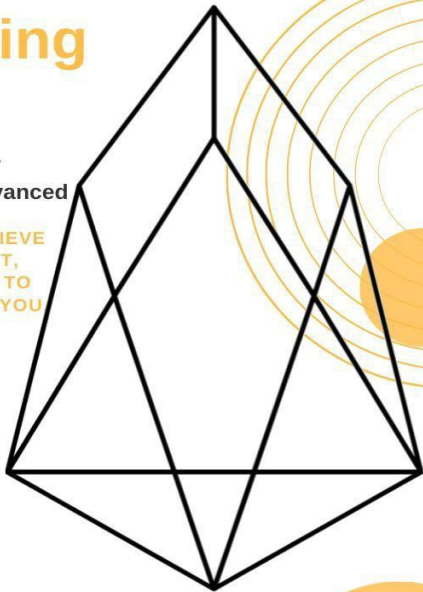


ALFREDO A. DE CANDIA

Mastering EOS

Practical guide for
beginners and advanced

"IF YOU DON'T BELIEVE
ME OR DON'T GET IT,
I DON'T HAVE TIME TO
TRY TO CONVINCE YOU
SORRY."



E O S

**ALFREDO
ANTONIO DE
CANDIA**

Mastering EOS

Guida pratica per principianti e
non

Prefazione

Scrivere un libro sulla crypto moneta EOS

Come tutti gli early adopter, in molti videro ed intuirono la potenza, di quello che sarebbe arrivato fino ai giorni nostri dopo 10 anni di vita, del protocollo blockchain e della prima crypto valuta legata ad essa ossia il Bitcoin, io dall'altro canto iniziai a vedere il progetto dalle retrovie, cercando di

capire il più possibile e di sperimentare questo nuovo sistema dove poi alla fine riuscii a creare il mio full node e a mantenerlo attivo dal 2011 al 2013, poi ovviamente con l'evolversi della rete non mi era più possibile continuare a supportare la cosa e quindi a dicembre di quell'anno staccai il tutto e di ritornare ad osservare il progetto e su come si sarebbe evoluto da lì in poi.

Passarono gli anni e vidi che il progetto adesso era abbastanza solido ma che questo comportava, a livello di mining amatoriale, delle soglie troppo elevate per dedicarmi come una volta e quindi puntai su sistemi "faucet" che danno Bitcoin per determinate operazioni, molto pochi ma che

comunque nel lungo periodo sono sempre utili.

Poi con l'avvento delle ICO (Initial Coin Offering) iniziarono a spuntare i primi progetti, alcuni interessanti altri delle vere e proprie truffe, e dove mi informai sommariamente su come si sarebbero evoluti in futuro, ammetto che all'inizio non ci avrei scommesso molto, anche perché bisognava investire non poco per avere qualcosa di interessante come profitto e quindi lasciai passare la cosa, ma ripromettendomi di controllare successivamente a progetto avviato.

Purtroppo in quel periodo passai un periodo leggermente orribile e quindi

vedevo abbastanza in bianco e nero il tutto e non mi resi conto di quello che stava succedendo, poi l'anno successivo, nel 2018, decisi di riprendere quello che avevo perso fino a quel momento e di recuperare terreno e quindi una volta pronto preparai il tutto per puntare la prossima scommessa sulla blockchain di EOS che si culminò con l'acquisto dei relativi token, dando fondo ai miei risparmi.

Ne rimasi subito colpito perché combinava la velocità ed un utilizzo pratico e diverso da quello che ero abituato a vedere fino a quel momento e poi con le varie dApp è stato veramente fantastico gestire il tutto tramite lo stesso account senza bisogno ogni volta

di spostare informazioni o fondi da un applicazione all'altra.

Da quel momento in poi mi sono veramente impegnato nei confronti di quella blockchain, ritenuta da molti imperfetta e che potrebbe fallire da un giorno all'altro, ovviamente c'è sempre questo rischio, però dal mio piccolo punto di vista scartare a priori qualcosa basandosi solo sul whitepaper o sul suo creatore, non so quanto possa essere costruttiva come obiezione, quindi nonostante gli "esperti" massacrano questo progetto io, al contrario gli do una possibilità e da quel poco che ho visto, ammetto anche io che ci sono problemi enormi nel progetto, cercherò nelle mie possibilità di aggiustare

alcune cose, per questo ho fatto diverse proposte di referendum sulla blockchain stessa per cercare di migliorare alcuni aspetti che potrebbero beneficiare tutta la comunità.

Pubblico di riferimento

Questo piccolo libro ha come scopo quello di informare e formare, dalla

persona comune allo sviluppatore interessato al progetto, un quadro storico, operativo e pratico dell'intero ecosistema, anche perché la carenza di informazioni, soprattutto in ambito europeo, per non parlare di quello italiano, è lacunoso e quindi qualcuno dovrebbe fare il primo passo e di conseguenza spero che questo possa portare altri autori a seguire la mia strada e diffondere informazioni, cultura e tecnologia.

Spiegazione del sottotitolo

Ho scelto la frase "IF YOU DON'T BELIEVE ME OR DON'T GET IT, I

DON'T HAVE TIME TO TRY TO CONVINCENOTE YOU, SORRY." perché oltre ad essere stata fatta dal presunto autore del Bitcoin, dimostro come potrei convincere qualcuno a seguire questo progetto, dandogli gli strumenti per valutare in maniera oggettiva tutta la questione esponendo tutte le critiche alla stessa e permettendogli di utilizzarla ai loro scopi e valutarla se sarà alle loro aspettative oppure dopo la lettura, i lettori, potranno regalare questo libro alla biblioteca della propria città.

Esempi di codice

Tutto il codice a cui si fa riferimento e

per le parti dettagliate è stato lasciato in inglese, volutamente, perché sono comandi pratici e noti a chi opera nel settore dello sviluppo e programmazione software e che comunque sono termini molto utilizzati in questo ambito.

Riferimento al codice e programma che si è utilizzato

Parte del codice che è stato inserito in questo libro è possibile recuperarlo nella pagina sviluppatori del portale EOS e muoversi tra i vari menù delle relative sezioni:

<https://developers.eos.io/eosio-home/docs>

Usare gli esempi di questo libro

Questo libro si pone come strumento per aiutare l'utente in diversi aspetti e gli esempi mostrati possono essere utilizzati da chiunque, purché citando questo libro come fonte di riferimento.

Indirizzi e transazioni relative a questo libro

Tutti gli indirizzi e le relative transazioni sono a fini istruttivi e quindi non sarebbe saggio utilizzare gli stessi

parametri perché ci potrebbe sempre essere un male intenzionato che possa approfittare di queste informazioni e qualsiasi trasferimento ai relativi account, volontario o meno, sarà interpretata come una semplice donazione, quindi non mandate nessuna transazione o token ai relativi account scritti in questo libro.

Dedicato alla mia Miss, che mi sopporta
ogni giorno.

Mastering EOS

Guida pratica per principianti e
non

Alfredo A. de Candia

Capitolo 1

Introduzione

Cos'è EOS

E' una infrastruttura potente decentralizzata per le applicazioni, basato su blockchain, ed è un sistema decentralizzato che permette lo sviluppo, l'hosting e l'esecuzione di applicazioni decentralizzate commerciali su larga scala (chiamate dApp) sulla sua piattaforma.

EOS permette sia agli imprenditori

che singoli individui di creare una dApp basata su blockchain in un modo simile a come si sviluppa un applicazione web, fornendo un accesso sicuro con relativa autenticazione, vari permessi, l'host di dati, la gestione e la comunicazione tra le varie dApp ed Internet.

La blockchain di EOS si pone com obiettivo di creare un sistema operativo decentralizzato su dove far girare le varie dApp ed inoltre ha il vantaggio di non avere fee per le transazioni e di gestire migliaia di transazioni al secondo (tps).

Un po' di storia di EOS

Tutto nasce dal whitepaper del 2017, avendo come obiettivo di creare un sistema altamente scalabile da poter gestire milioni di transazioni, dove tutta la piattaforma è stata sviluppata da block.one che ha rilasciato il software open source il primo giugno 2018, di fatto il compleanno del progetto, passando dalla fase di test iniziata il 3 settembre del 2017.

Il logo di EOS deriva dal Chestahedron (eptaedro), ossia una figura geometrica creata da Frank Chester^[1], artista con un'impronta platonica dato che si ispira ai solidi platonici (il tetraedro regolare, l'esaedro regolare, l'ottaedro regolare, il

dodecaedro regolare e l'icosaedro regolare) che per gli antichi greci rappresentavano i 4 elementi, terra, aria, fuoco e acqua, ed il quinto la forza vitale o spirito.

Il Chestahedron, a differenza dei solidi platonici che hanno una sola faccia, ha 2 tipi di facce ossia triangoli ed aquiloni, ma da notare che tutte le facce, 7, hanno la stessa quantità di superficie^[2] e sono 4 triangoli e 3 quadrilateri, dove contiene 7 punti e 12 spigoli, dove per il creatore di questa figura rappresenta il cuore.

Per garantire una vasta distribuzione del token, l'azienda, ha distribuito 1 miliardo di token ERC20 a coloro che registravano un indirizzo Ethereum e

dove poi successivamente avrebbero reclamato i relativi token EOS.

Dietro al progetto ci sono Daniel Larimer CTO dell'azienda ed anche creatore della delegated proof-of-stake (DPoS) e Brandan Blumer, attuale CEO di block.one.

Il progetto ha sfruttato il sistema delle ICO (Initial COin Offer) basato sulla blockchain di Ethereum, dove il 26 giugno 2017 è partita la vendita e si è conclusa il 3 luglio 2018, in 350 round giornalieri da 2 milioni di token ed al prezzo di mercato. Infatti gli EOS token erano un token ERC20 distribuito sulla blockchain di Ethereum dove il 20% del totale, 200 milioni di EOS token sono stati distribuiti in 5 giorni dal 26 giugno

2017 al primo luglio 2017 e dopo quella data sono stati distribuiti 700 milioni di EOS token, mentre invece 100 milioni di EOS token sono rimasti a block.one.

La conversione da Ethereum a EOS ha seguito questo calcolo:

$$\text{EOS ricevuti} = a * (b/c)$$

- a rappresenta il totale degli ETH contribuiti da un acquisto autorizzato durante quel periodo;
- b rappresenta il totale dei token EOS disponibili per la distribuzione in quel

periodo;

- c rappresenta il totale ETH che hanno contribuito da tutti gli acquisti autorizzati durante quel periodo.

Caratteristiche della blockchain di EOS

Scalabilità:

Il problema principale che affligge ogni blockchain è quello relativo alla scalabilità, ossia gestire un numero elevato di transazioni, ma che con i protocolli di consenso che ci sono determinano che ogni nodo si metta

"d'accordo" per procedere con il tutto, mentre con la DPoS, distribuisce questo meccanismo di consenso riuscendo a gestire milioni di transazioni.

Flessibilità:

Il sistema, ovvero i produttori di blocchi (Block Producer, BP) permette di bloccare una dApp dannosa o che crea problemi, fino a quando il sistema non gestirà la cosa, e dove si impedisce che la rete ne risenta ed inoltre, tramite la DPoS non è necessario che ogni nodo si occupi della manutenzione della rete.

Usabilità:

EOS permette livelli ben definiti di permesso, incorporando caratteristiche come i web toolkit per l'interfaccia da sviluppare, interfacce auto-descrittive, schemi di database auto-descrittivi e uno schema di permesso dichiarativo.

Governance:

La governace di EOS è data dal mutuo riconoscimento ed accettazione di regole che sono scritte nella sua costituzione, esatto la blockchain di EOS ha una costituzione propria, e dove ogni transazione in EOS deve includere obbligatoriamente l'hash della costituzione nella firma, legando in

questo modo l'utente alla costituzione.

Processi paralleli:

Nei processi paralleli, le istruzioni sono divise tra diversi processori ed in questo modo il tempo per far girare il tutto diminuisce in maniera elevata, dove EOS fornisce un processo parallelo degli smart contract tramite la scalabilità orizzontale (aggiungendo più sistemi e più computer alla pool di risorse), la comunicazione asincrona (le parti coinvolte non devono essere presenti nello stesso momento per comunicare) e l'interoperabilità

(possibilità ad ogni computer di scambiare ed utilizzare informazioni).

Auto sufficienza:

Ogni sistema basato sul software di EOS deve generare una percentuale di inflazione all'anno, dove questa sarà distribuita dalla piattaforma ai BP in proporzione alla conferme delle transazioni sulla piattaforma stessa , ed ai miglior 3 smart contract o proposte che hanno ricevuto il maggior numero di

voti dagli holder di quel token. In questo modo si garantisce l'auto sufficienza della blockchain da un appoggio esterno come fondazioni, organizzazioni o singolo individui, per la sua crescita, sviluppo e mantenimento.

Sistema operativo decentralizzato:

Quindi come un tradizionale sistema operativo, questo gestisce tutto il funzionamento e dove il protocollo emula le caratteristiche hardware tipiche di un computer reale ossia CPU E GPU per l'elaborazione dei dati, RAM per il salvataggio di alcuni dati e NET, che è la banda di rete relativa allo stesso e

dove la potenza computazionale è distribuita equamente tra i possessori della crypto moneta EOS, dopo averli messi in stake (cioè vincolati).

Capitolo 2

Come iniziare

Per poter operare sulla blockchain di EOS è richiesto un account, un nome leggibile nel linguaggio umano, che è salvato all'interno della blockchain, può appartenere ad un individuo o ad un gruppo di individui a seconda del permesso dello stesso, e questo account è necessario per effettuare tutte le operazioni sulla blockchain.

Ogni account EOS ha una lunghezza di 12 caratteri (salvo non sia un account premium) che contiene i caratteri dalla a

alla z (non si possono utilizzare lettere maiuscole) e cifre da 1 a 5 (lo 0 non è ammesso).

Un account EOS consiste in 2 chiavi, la chiave attiva (active key) e la chiave proprietaria (owner key), dove la chiave attiva può essere usata per trasferire fondi, votare per i BP, comprare RAM e così via, mentre la chiave proprietaria, dimostra la proprietà dell'account ed è richiesta per fare qualsiasi tipo di modifica riguardo la proprietà dell'account, dato che si può cambiare la proprietà dello stesso, inoltre questa chiave è meglio tenerla offline dato che per la maggior parte delle operazioni non è richiesta il suo utilizzo.

Come creare account EOS

Ci sono diversi metodi per creare un account EOS, tramite app per cellulari, tramite servizi web, tramite un altro account EOS e così via.

La cosa importante quando si crea un account EOS è quella di procurarci le chiavi per il nostro account, queste o le generiamo noi tramite apposite applicazioni come Scatter o servizi

come EOSKEY^[3], oppure vengono generate con la creazione del nuovo account e poi dovremmo salvare quelle chiavi.

Per esempio con Scatter possiamo creare una coppia di chiavi, la owner key e la active key dove entrambe hanno la loro chiave privata, che saranno indispensabili per creare il nuovo account.

Per esempio possiamo usare il servizio di eos account creator^[4] dove ci guiderà passo dopo passo nella creazione dell'account, permettendo inoltre di comprare le risorse indispensabili per il nostro account come la RAM:

- una volta collegati sul sito bisogna

clikcare il pulsante "create eos account";
- nella pagina successiva dovremmo scegliere il nome del nostro account, considerando che potremmo incontrare il problema che l'account scelto sia già stato preso e quindi dovremmo sceglierne un altro (c'è una spunta verde quando l'account è disponibile) e poi clicchiamo su continua;

- in questa finestra dobbiamo inserire le precedenti chiavi generate tramite Scatter o altro sistema (in questa pagina è possibile anche recuperare altri metodi per generare le nostre chiavi), oppure possiamo connettere anche un nostro ledger (dispositivo elettronico che conserva le nostre chiavi private);

- nella pagina successiva dobbiamo

scegliere il metodo di pagamento, carta di credito, crypto monete (Bitcoin, Bitcoin Cash, Ethereum o Litecoin) oppure EOS se per caso ne abbiamo in qualche Exchange, per continuare l'operazione, e a seconda del metodo seguiamo le istruzioni a video:

EOS

Account creator

This website allows you to **create an EOS account** if you don't have one yet. [Check our FAQ](#) for more info.

New! We are now offering a tool to [buy EOS RAM](#) with Credit Card or Cryptocurrency (Bitcoin, Ethereum, Bitcoin Cash, Litecoin). If you're getting error messages like *Account using more than allotted RAM usage* or messages like *account has insufficient ram; needs 8192 bytes has 4096 bytes*, that means you have run out of RAM and need to *buy more EOS RAM*. [Buy more EOS RAM now!](#)

Why this service?

The way EOS works is that new accounts can only be created by someone with an existing account. Creating an account also requires to stake a certain amount of EOS and to buy some RAM.

If you create an EOS account on this website, it comes with 0.2 EOS staked for CPU and Network bandwidth as well as 4KB of RAM.



[CREATE EOS ACCOUNT](#)

01

Choose account name

Please choose your EOS Account Name. [Check our FAQ](#) for more info.

Choose account name

EOS Account names must be exactly 12 characters long and consist of lower case characters and digits up until 5.

Account name

[I'm feeling lucky](#)



[CONTINUE](#)

02

Provide public keys

Your chosen EOS Account Name is: [robberto1234](#)

What are those keys and how do I generate them?

In order to access your new EOS account, you will need two key pairs, owner and active. Each key pair consists of a private key and a public key. The public key is safe to give out and will be publicly visible in the blockchain. The private key must be kept secret and stored securely.

Here are several generators you can use:

- [Simple Javascript generator](#) (easiest to use)
- [EOSKEY](#) (offline key generator, easy to use, but no wallet included)

Recommended wallets:

- [Infinito Wallet](#) World's leading universal wallet with eos-account-creator being natively integrated for seamless user experience
- [TokenPocket](#) The biggest DApp Store on EOS, you can play hundreds of DApps and chat with players here
- [Greymass Wallet](#) (Fully featured desktop wallet)
- [SimpleEOS](#) (Easy to use desktop wallet, supports generating keys offline)
- [Scatter](#) (Chrome Desktop, recommended for daily use)
- [Cleos](#) (Official wallet by Block.one but command line tool, only for advanced users)

Provide public keys

Please provide the public keys for your new account. Active Public Key can be the same as Owner Public Key but not recommended for security.

Owner Public Key:

[Connect Ledger](#)

Active Public Key:

[Continue](#)

03

Payment

Your chosen EOS Account Name is: [robberto1234](#)

Owner Public Key:

Active Public Key:

Please consider voting for [eosvibesbloc](#)

One last step

After successful payment you will be redirected back here and we will create the EOS account for you. If you have any questions, please visit our [Telegram group](#)

Country of residence

Italy

Choose payment option

EOS (Use this if you have EOS on an exchange)

 **0.8 EOS**

When paying with EOS

I have read and accept the [privacy policy](#) and the EOS Constitution

[Show instructions](#)

EOS Smart Contract Account Creator

[EOS Account Creator](#) is proud to present the world's first [open source](#) smart contract account creator. You can use it to create an account when you have EOS on an exchange.

Instructions

Make an EOS token transfer with the following data:

Recipient: accountcreat

Memo: robberto2345vdFPuVfs4d 

Amount: Any amount, but has to be bigger than 0.7908 EOS

If you transfer less than the required minimum amount or any other error occurs, the transferred funds will be automatically refunded by the smart contract.

If you transfer more than required, the remaining balance will be forwarded to your newly created account.

Dopo che completeremo l'operazione nel giro di alcuni minuti il nostro account sarà operativo e pronto all'uso.

Possiamo creare anche un account EOS partendo da un altro account EOS ed in questo caso a seconda dello strumento utilizzato cambierà l'interfaccia grafica ma la sostanza non cambia, infatti come nella procedura di prima l'unica cosa in più che potremmo decidere è il totale delle risorse da assegnare al nuovo account (NET, CPU e RAM), nell'esempio successivo

vediamo la procedura tramite il sito eosx.io^[5].

 ACCOUNT

CREATE ACCOUNT

- Create account via exchange withdrawal or help from others
- Create account using an existing account's funds (Requires Scatter)

Creator Name (Linked to your Scatter)

Account Name that will create the new Account

New Account Name

justfortests

12 characters using a-z and 1-5 only

Active Key

Public key used for most actions

53 characters, begins with 'EOS'

Owner Key

Public key that will own the account

53 characters, begins with 'EOS'

NET Stake (in EOS)

0,2

CPU Stake (in EOS)

0,2

RAM (in bytes)

4096

Transfer Stake To New Account

If checked, the stake belongs to newly created account. If unchecked, the stake belongs to the creator.

At current RAM prices, the total cost will be about **0.6525 EOS** = \$1.42

CREATE ACCOUNT

Creare account tramite Cleos

E' possibile creare anche un account scrivendo direttamente il codice utilizzando cleos ed in questo caso bisogna partire prima dalla creazioni delle chiavi, quella pubblica e quella privata.

create key

Positionals:

none

Options:

--r1 - Generate a key using the R1 curve (iPhone), instead of the K1

curve (Bitcoin)

-f, --file *TEXT* - Name of file to write private/public key output to. (Must be set, unless "--to-console" is passed.

--to-console - Print private/public keys to console.

Usage:

```
$ ./cleos create key -f myKey.txt
```

```
$ ./cleos create key --to-console
```

Output:

Private key:

```
5KCkcSxYKZfh5Cr8CCunS2PiUKzNZL
```

Public key:

EOS5uHeBsURAT6bBXNtvwKtWaiDSl

Poi si passa alla creazione del nuovo account sulla blockchain (assumendo che non ci sono limiti per quanto riguarda l'utilizzo della RAM, dato che comunque serve un poco di RAM per poter creare un nuovo account).

create account

Positionals:

creator *TEXT* - The name of the account creating the new account

name *TEXT* - The name of the new account

OwnerKey *TEXT* - The owner

public key or permission level for the new account (required)

ActiveKey *TEXT* - The active public key or permission level for the new account

Options:

-h,--help - Print this help message and exit

-x,--expiration - Set the time in seconds before a transaction expires, defaults to 30s

-f,--force-unique - force the transaction to be unique. this will consume extra bandwidth and remove any protections against accidentally issuing the same transaction multiple

times

-s,--skip-sign - Specify if unlocked wallet keys should be used to sign transaction

-j,--json - Print result as json

-d,--dont-broadcast - Don't broadcast transaction to the network (just print to stdout)

--return-packed - Used in conjunction with --dont-broadcast to get the packed transaction

-r,--ref-block *TEXT* - Set the reference block num or block id used for TAPOS (Transaction as Proof-of-Stake)

-p,--permission *TEXT* - An account and permission level to

authorize, as in 'account@permission'
(defaults to 'creator@active')

`--max-cpu-usage-ms` *UINT* - Set an upper limit on the milliseconds of cpu usage budget, for the execution of the transaction (defaults to 0 which means no limit)

`--max-net-usage` *UINT* - set an upper limit on the net usage budget, in bytes, for the transaction (defaults to 0 which means no limit)

`--delay-sec` *UINT* - Set the `delay_sec` seconds, defaults to 0s

Usage:

Usage: cleos create account [OPTIONS]

creator name OwnerKey [ActiveKey]

Positionals:

creator TEXT The name of
the account creating the new account
(required)

name TEXT The name of
the new account (required)

OwnerKey TEXT The owner
public key or permission level for the
new account (required)

ActiveKey TEXT The active
public key or permission level for the
new account

Options:

-h,--help Print this help message and exit

-x,--expiration set the time in seconds before a transaction expires, defaults to 30s

-f,--force-unique force the transaction to be unique. this will consume extra bandwidth and remove any protections against accidentally issuing the same transaction multiple times

-s,--skip-sign Specify if unlocked wallet keys should be used to sign transaction

-j,--json print result as json

-d,--dont-broadcast don't broadcast transaction to the network (just print to stdout)

`--return-packed` used in conjunction with `--dont-broadcast` to get the packed transaction

`-r,--ref-block TEXT` set the reference block num or block id used for TAPOS (Transaction as Proof-of-Stake)

`-p,--permission TEXT ...` An account and permission level to authorize, as in 'account@permission' (defaults to 'creator@active')

`--max-cpu-usage-ms UINT` set an upper limit on the milliseconds of cpu usage budget, for the execution of the transaction (defaults to 0 which means no limit)

`--max-net-usage UINT` set an upper limit on the net usage budget, in bytes, for the transaction (defaults to 0 which

means no limit)

`--delay-sec` UINT set the
delay_sec seconds, defaults to 0s

Command:

Un set di chiavi è necessario per creare un account e può essere creato utilizzando `./cleos create key` using public keys

```
$ ./cleos create account inita tester  
EOS4toFS3YXEQCkuuw1aqDLrtHim86  
EOS7d9A3uLe6As66jzN8j44TXJUqJSK
```

using permission levels

```
$ ./cleos create account eosio
```

eosio.token eosio@active eosio@active

Output:

```
{
    "transaction_id":
"6acd2ece68c4b86c1fa209c3989235063
"processed": {
    "refBlockNum": "25217",
    "refBlockPrefix": "2095475630",
    "expiration": "2017-07-
25T17:54:55",
    "scope": [
        "eos",
        "inita"
    ],
    "signatures": [],
    "messages": [{
```


}

E' possibile creare anche un account partendo da un account esistente ed in questo caso serve un account esistente, un username idoneo e 2 paia di chiavi per avere una maggiore sicurezza.

system newaccount

Positionals:

creator *TEXT* - The name of the account creating the new account

name *TEXT* - The name of the new account

OwnerKey *TEXT* - The owner public key for the new account

ActiveKey *TEXT* - The active public key for the new account

Options:

-h,--help Print this help message and exit

--stake-net *TEXT* - The amount of EOS delegated for net bandwidth

--stake-cpu *TEXT* - The amount of EOS delegated for CPU bandwidth

--buy-ram-kbytes *UINT* - The amount of RAM bytes to purchase for the new account in kibibytes (KiB), default is 8 KiB

--buy-ram *TEXT* - The amount of RAM bytes to purchase for the new account in EOS

--transfer - Transfer voting

power and right to unstake EOS to receiver

-x,--expiration *TEXT* - set the time in seconds before a transaction expires, defaults to 30s

-f,--force-unique - force the transaction to be unique. this will consume extra bandwidth and remove any protections against accidentally issuing the same transaction multiple times

-s,--skip-sign Specify if unlocked wallet keys should be used to sign transaction

-d,--dont-broadcast - Don't broadcast transaction to the network (just print to stdout)

`-r,--ref-block TEXT` set the reference block num or block id used for TAPOS (Transaction as Proof-of-Stake)

`-p,--permission TEXT` - An account and permission level to authorize, as in 'account@permission' (defaults to 'account@active')

`--max-cpu-usage-ms UINT` - set an upper limit on the milliseconds of cpu usage budget, for the execution of the transaction (defaults to 0 which means no limit)

`--max-net-usage UINT` - set an upper limit on the net usage budget, in bytes, for the transaction (defaults to 0 which means no limit)

Usage:

```
cleos system newaccount eosio  
someusername EOS123456..  
EOS123456...
```

Account EOS premium

Abbiamo detto che la maggior parte degli account EOS hanno 3 parametri standard di default:

- sono lunghi 12 caratteri;
- possono contenere solo lettere minuscole dalla a alla z;
- ed avere un numero compreso dall'1 al 5.

Ma comunque c'è un modo per avere

account ancora più corti o con una determinata estensione (questi nomi includono il suffisso del punto "."), grazie ad una proposta fatta proprio da Daniel Larimer (<https://github.com/EOSIO/eos/issues/31>) e dove ogni giorno viene messo a disposizione, tramite un'asta al migliore offerente, solo 1 account premium, ma se viene effettuata una nuova offerta nell'arco delle 24 ore allora non ci sarà nessun nome dato e si aspetterà che finisca prima la precedente asta.

Chi vince l'asta , inoltre, potrebbe decidere di vendere i nomi che vengono generati con quel tipo di struttura e quindi essere utile per alcuni persone o aziende, pensiamo all'estensione ".com",

dove un'azienda potrebbe comprare il nome "azienda.com" e quindi avere un nome coerente con il suo ecosistema.

Chiunque può partecipare a queste aste e fare offerte si possono utilizzare diversi strumenti, oppure tramite cleos è possibile usare il comando: `bidname (account_name bidder, account_name desired, asset bid)`, che hanno la relativa sezione "bid name" o "premium name" e poi, sempre dopo esserci collegati con il nostro account EOS, possiamo inserire il nome che vogliamo, inseriamo il nostro account ed infine l'importo massimo che vogliamo offrire per quel nome e per fare un'offerta bisogna puntare almeno il 10% in più rispetto all'offerta più alta, e dove i

relativi eos saranno trasferiti ed utilizzati se ci aggiudichiamo l'asta, altrimenti al termine della stessa, in caso di esito negativo, ci verranno restituiti i nostri eos, quindi di fatto i nostri eos saranno congelati per diverso tempo fino a quando non verrà completata, ricordiamo si completa un'asta al giorno quindi il tempo è veramente lungo.

Name Bids

Current Bids

Need to be Claimed

Sold Names

Your Account Name

Name to Bid On

Amount to Bid (in EOS)

Bidder - Login to wallet

Name to Buy

e.g. 10000

Bid

Name	Bidder	Amount	Bid Time (CEST)
up	z.io	500 EOS	Jul 15, 02:50:06 PM
fc	gy2tombsgege	466 EOS	Jul 13, 11:46:13 AM
run	mytoken.best	458 EOS	Jul 12, 10:57:01 AM

Current Bids

Need to be Claimed

Sold Names

Name	Bidder	Amount	Creation TX	Bid Time (CEST)
av	sunxiaoliang	660.0001 EOS	deabf566...	Jul 13, 02:46:06 PM
do	gm2tkmbyguge	503 EOS	f171603d...	Jul 12, 01:09:47 PM
ea	surprisecase	456 EOS	046eea3d...	Jul 09, 08:13:17 PM
dj	z.io	457 EOS	1f6afc53...	Jul 11, 08:14:02 AM
best	kindleforest	760 EOS	75dd78e6...	May 25, 04:24:42 PM

eosio - bidname

bid: 660.0001 EOS
 bidder: sunxiaoliang
 newname: av

Transfer - eosio.token

sunxiaoliang → eosio.names 660.0001 EOS

Memo: bid name av

Piazzare offerte per account premium con Cleos

system bidname

Positionals:

bidder *TEXT* - The bidding account (required)

newname *TEXT* - The bidding name (required)

bid *TEXT* - The amount of CORE SYMBOL to bid (required)

Options:

-h,--help - Print this help message and exit

-x,--expiration - set the time in seconds before a transaction expires, defaults to 30s

-f,--force-unique - force the transaction to be unique. this will consume extra bandwidth and remove any protections against accidentally issuing the same transaction multiple times

-s,--skip-sign - Specify if unlocked wallet keys should be used to sign transaction

-j,--json - print result as json

-d,--dont-broadcast - don't broadcast transaction to the network (just print to stdout)

-r,--ref-block *TEXT* - set the

reference block num or block id used for TAPOS (Transaction as Proof-of-Stake)

-p,--permission *TEXT* - An account and permission level to authorize, as in 'account@permission'

--max-cpu-usage-ms *UINT* - set an upper limit on the milliseconds of cpu usage budget, for the execution of the transaction (defaults to 0 which means no limit)

--max-net-usage *UINT* - set an upper limit on the net usage budget, in bytes, for the transaction (defaults to 0 which means no limit)

Usage:

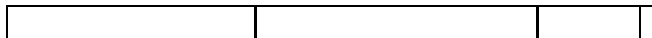

```
cleos system bidname accountname1 bob  
"10 SYS"
```

Account con multi firma

Una differenza principale di EOS rispetto alle altre blockchain è che utilizza un sistema di account, composto normalmente da 12 caratteri, con cui si interagisce con la blockchain e se vogliamo utilizzare e spendere i nostri fondi, dobbiamo firmare la transazione con il nostro account e quindi non con la nostra chiave direttamente, infatti dietro ad un account ci possono essere una o più chiavi che ovviamente sono sulla

blockchain e dove le possiamo cambiare oppure settare diversi livello di permesso.

Di solito ogni account ha di default 2 chiavi, quella proprietaria e quella attiva; dato che è possibile cambiare i permessi di un account EOS è anche possibile creare un account con una multi firma, quindi rendere necessario più di una persona per confermare una determinata transazione per l'account stesso.



Permesso chiave	Account	Peso
proprietario		
	@roberto1234	1
	@giuseppe123	1
attiva		
	@roberto1234	1
	@giuseppe123	1
pubblicazione		
	@roberto1234	2
	@giuseppe123	2
	@marcello1234	11

Nel primo esempio per effettuare delle operazioni all'account proprietario è necessario che tutti gli account devono confermare ed autorizzare il

cambiamento (in quel caos la soglia è 2);

Nel secondo esempio, qualsiasi operazione che richiede la chiave attiva, può essere effettuata da un solo account senza che necessiti il permesso dell'altro account per effettuare le operazioni;

Nel terzo esempio vediamo che ci sono più account ma che solo alcuni hanno il peso (2) per operare indipendentemente dagli altri, in questo caso pubblicare qualcosa, mentre invece c'è un account, @marcello1234, che ha un peso inferiore e quindi serve il permesso e quindi la firma di uno degli account che hanno questa possibilità.

Come si crea un account multi firma

La procedura di seguito utilizzerà il servizio di EOSToolkit. Per prima cosa dobbiamo collegarci alla pagina e poi collegare il nostro account EOS tramite il programma Scatter^[6]:



This action has serious consequences - You can make your account IRRECOVERABLE

EOS accounts can have complex permission structures which include a parent/child relationship. Every account starts with the basic structure `owner`, which is the parent permission for all future permissions, and it's child `active`.

Parent permissions can always change or remove the child permissions. For example, `active` can have the child permission `delegate` added beneath it. Both `owner` and `active` can change or remove the `delegate` permission because it is the child of both.

```
owner: { keys,accounts,delays... }
```

```
  active: { keys,accounts,delays... }
```

```
    delegate: { keys,accounts,delays... }
```

Each permission itself has a threshold, and can have a set of keys, accounts, or delays associated with various weight. If the threshold is 1, any of these authorities with a weight of 1 can execute a transaction. If the threshold is 2 you will require the signatures of two `weight 1` authorities, or a single `weight 2` authority.

Adding or modifying permissions

Threshold is the required sum of permission weights to execute an action.

Permission is the name of the new permission.

Parent is the parent permission of the new permission.


Authority can be an actor authority in the format `accountname@permission`, a public key, or a delay in seconds.

Weight is how much weight this Authority lends to the Threshold.

You can add or remove rows as required to meet your multisig requirements.

Remove a permission

Specify the Permission and Parent, and leave a single Authority row empty with the default Weight of 1.



Change Permissions - Advanced permission structures

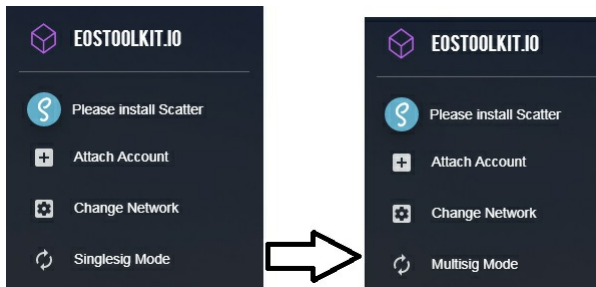
Account	Threshold
roberto1234	2
Permission	Parent
active	owner
Authority	Weight
roberto1234@permission	2
Authority	Weight
giuseppe123@permission	1

By executing this action you are agreeing to the EOS constitution and this action's associated ricardian contract. The ricardian contract may be viewed in the Scatter approval prompt.

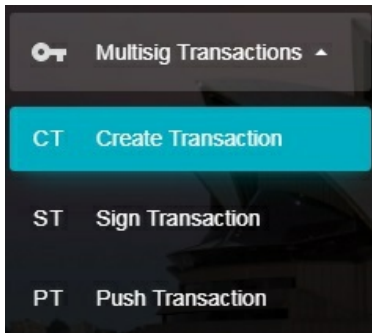
Un volta completati i parametri, aggiungendo o rimuovendo account, si procede con l'Update delle informazioni e bisognerà confermare il tutto tramite Scatter (si aprirà una finestra pop-up).

Dopo che abbiamo creato la multi firma, sempre tramite lo stesso sito web possiamo creare e firmare le

transazioni^[7], per prima cosa dobbiamo collegarci con Scatter e poi passare alla modalità multisig in alto a sinistra:



Poi a seconda dell'operazione che dobbiamo fare ci rechiamo nel menù multisig transactions:



Poi nella finestra successiva andremo a completare i parametri con tutte le informazioni, nella creazione della transazione andremo a creare un file JSON che ci servirà per firmare la transazione dato che dovremmo caricare il file precedentemente creato per poterlo firmare ed infine nella sezione push andremmo a inserire tutto quello creato e firmato nella rete.

Settare permessi tramite Cleos

set account permission

Positional:

account *TEXT* - The account to set/delete a permission authority for

permission *TEXT* - The permission name to set/delete an authority for

authority *TEXT* - [delete] NULL, [create/update] public key, JSON string, or filename defining the authority

parent *TEXT* - [create] The permission name of this parents permission (Defaults to: "Active")

Options:

-h,--help Print this help message and exit

--add-code [code] add 'eosio.code' permission to specified permission authority

--remove-code [code] remove 'eosio.code' permission from specified permission authority

-x,--expiration *TEXT* - set the time in seconds before a transaction expires, defaults to 30s

-f,--force-unique - force the transaction to be unique. this will

consume extra bandwidth and remove any protections against accidentally issuing the same transaction multiple times

-s,--skip-sign Specify if unlocked wallet keys should be used to sign transaction

-d,--dont-broadcast - Don't broadcast transaction to the network (just print to stdout)

-r,--ref-block *TEXT* set the reference block num or block id used for TAPOS (Transaction as Proof-of-Stake)

-p,--permission *TEXT* - An account and permission level to authorize, as in 'account@permission'

(defaults to 'account@active')

`--max-cpu-usage-ms` *UINT* - set an upper limit on the milliseconds of cpu usage budget, for the execution of the transaction (defaults to 0 which means no limit)

`--max-net-usage` *UINT* - set an upper limit on the net usage budget, in bytes, for the transaction (defaults to 0 which means no limit)

`-j,--json` Print result as json

Usage:

Per modificare il permesso di un account è necessario avere l'autorità di quell'account e il permesso di quello che si sta modificando.

Basic usage, set new key to a permission
\$./cleos set account permission
testaccount active EOSPUBLICKEY
owner -p testaccount@owner

Basic usage, set an account (instead of a
key) as authority for a permission.

\$./cleos set account permission
testaccount active diffaccount owner -p
testaccount@owner

Advanced Usage, weight/thresholds

\$./cleos set account permission
testaccount active '{"threshold" : 100,
"keys" : [], "accounts" : [{"permission":
{"actor": "user1", "permission": "active"},
{"permission":
{"actor": "user2", "permission": "active"}},
owner -p testaccount@owner

L'oggetto JSON usato in questo comando

è attualmente composto da due tipi differenti di oggetti.

L'oggetto JSON authority contiene:

```
{  
  "threshold"      : 100,    /*An integer  
that defines cumulative signature weight  
required for authorization*/  
  "keys"           : [],     /*An array made  
up of individual permissions defined  
with an EOSIO-style PUBLIC KEY*/  
  "accounts"       : []     /*An array made  
up of individual permissions defined  
with an EOSIO-style ACCOUNT*/  
}
```

Returns:

contiene 0 o più oggetti nell'array della

chiave

```
{  
    "key" :  
"EOS8X7Mp7apQWtL6T2sfSZzBcQNU  
    "weight" : 25 /*Set the weight  
of a signature from this permission*/  
}
```

contiene 0 o più oggetti nell'array
dell'account

```
{  
    "permission" : {  
        "actor" : "sandwich",  
        "permission" : "active"  
    },  
    "weight" : 75 /*Set the weight of  
a signature from this permission*/  
}
```


Come scegliere il wallet EOS

In generale esistono 3 principali strumenti, wallet, che permettono di gestire e salvare le nostre chiavi di EOS:

- tramite wallet fisici (hardware), che sono dei dispositivi pensati per salvare le chiavi private dei diversi indirizzi, quindi supportano anche altre blockchain, che permettono di firmare le transazioni senza che la chiave privata venga esposta a terzi con un'interfaccia minimale con dei pulsanti per confermare il tutto, da notare che per alcune dApp di EOS non è possibile collegare, ancora, un ledger fisico per

confermare le transazioni;

- wallet software, programmi su desktop o mobile che permettono di gestire appieno tutto l'ecosistema di EOS e quindi utilizzare la nostra chiave, di solito si inserisce solo la chiave attiva così da non dover usare quella proprietaria, per poter collegarsi e gestire le varie dApp in maniera efficace e confermare ogni volta le varie transazioni, mostrando tutti i punti salienti del relativo smart contract;

- wallet cartacei (paper wallet), che, tramite alcuni servizi online, permettono di generare sia la chiave privata che la chiave pubblica sotto forma di codice QR, che poi andremo a stampare, prendendo tutte le precauzioni relative, e

la conserveremo al sicuro, ovviamente questo metodo serve solo per salvare offline quelle informazioni dato che non sarebbe comodo come utilizzo;

- servizi online che permettono di gestire il wallet direttamente da un browser e quindi essere funzionale su qualsiasi dispositivo con una connessione

Come compreso l'utilizzo ottimale per l'ecosistema EOS è quello si utilizzare un software desktop o mobile, quello principale per il desktop è senz'altro Scatter, un wallet con funzionalità diverse, o wallet richiesto, se non da tutti, dalla maggior parte delle dApp per poter confermare le varie transazioni; mentre su mobile ci sono

diverse applicazioni sia per android che ios che permettono di gestire all'interno dell'app stessa tutto il sistema EOS e quindi interagire con le dApp, ed inoltre offrono anche layer di sicurezza come pin e protezione biometrica, se presente nel dispositivo stesso.

Creare e gestire un wallet con Cleos

create

E' possibile creare un wallet anche tramite linea di comandi e specificare il nome del wallet stesso e se nessun nome viene scelto allora questo sarà "default".

Positionals:

none

Options:

-n, --name *TEXT* - The name of the new wallet

-f, --file *TEXT* - Name of file to write wallet password output to. (Must be set, unless "--to-console" is passed)

--to-console - Print password to console

Usage:

```
$ ./cleos wallet create --to-console
```

or

```
$ ./cleos wallet create -n second-wallet  
--to-console
```

or

```
$ ./cleos wallet create --name my-new-  
wallet --file my-new-wallet.txt
```

Outputs:

Creating wallet: default

Save password to use in the future to
unlock this wallet.

Without password imported keys will
not be retrievable.

```
"PW5JD9cw9YY288AXPvnbwUk5JK4C
```

oppure

Creating wallet: second-wallet

Save password to use in the future to unlock this wallet.

Without password imported keys will not be retrievable.

"PW5Ji6JUrLjhKAVn68nmacLxwhvtqU/

open

Con questa funzione si può aprire il wallet.

Positionals:

none

Options:

-n, --name *TEXT* - The name of the wallet to open.

Usage:

```
$ ./cleos wallet open
```

```
oppure
```

```
$ ./cleos wallet open -n second-wallet
```

Outputs:

Opened: default

lock

Tramite questa funzione si blocca il wallet.

Positionals:

none

Options:

-n, --name *TEXT* - The name of the wallet to lock

Usage:

\$ cleos wallet lock

oppure

```
$ ./cleos wallet lock -n second-wallet
```

Outputs:

```
Locked: 'default'
```

oppure

```
Locked: 'second-wallet'
```

lock_all

Con questa funzione si bloccano tutti I wallet aperti.

Positionals:

none

Options:

none

Usage:

\$./cleos wallet lock_all

Outputs:

Locked All Wallets

unlock

Sblocca un wallet.

Positionals:

none

Options:

-n, --name *TEXT* - The name of the wallet to unlock.

--password *TEXT* - The password returned by wallet create.

Usage:

Per sbloccare un wallet bisogna specificare la password che è stata utilizzata per crearlo

\$./cleos wallet unlock -n second-wallet

--password

PW5Ji6JU rLjhKAVn68nmacLxwhvtqUA

Outputs:

Unlocked: 'second-wallet'

import

Si importa la chiave privata all'interno del wallet.

Positionals:

none

Options:

`-n, --name TEXT` - The name of the wallet to import key into.

`--private-key TEXT` - Private key in WIF format to import.

Usage:

```
$ ./cleos wallet import --private-key  
5KQwrPbwdL6PhXujxW37FSSQZ1Jiws
```

Outputs:

```
imported      private      key      for:  
EOS6MRyAjQq8ud7hVNYcfnVPJqcVps
```

list

Crea una lista con tutti I wallet aperti, e

quelli che sono bloccati sono identificati con un asterisco *.

Positionals:

none

Options:

none

Usage:

\$./cleos wallet list

Outputs:

Wallets:

```
[  
  "default *",  
  "second-wallet *"  
]
```

oppure quando non ci sono wallet

Wallets:

```
[  
]  
]
```

keys

Crea una lista della chiavi pubbliche da tutti i wallet sbloccati, queste sono le chiavi che possono essere usate per

firmare le transazioni.

Positionals:

none

Options:

none

Usage:

\$./cleos wallet keys

Outputs:

[[

"EOS6MRyAjQq8ud7hVNYcfnVPJqcVp

```
"5KQwrPbwdL6PhXujxW37FSSQZ1Jiw  
]  
]
```

create_key

Crea un paio di chiavi all'interno del wallet così da evitare che si importino manualmente, dove di default creerà delle chiavi di tipo "facored" dal wallet, che è una chiave K1, ma è possibile creare anche chiavi in formato R1.

Positionals:

key_type *TEXT* - "K1" or "R1" Key type

to create

Options:

`-n,--name TEXT=default` The name of the wallet to create key into

Usage:

```
$ cleos wallet create_key K1
```

Outputs:

Created new private key with a public key of:

```
"EOS67xHKzQArkWZN6rKLCq7NLvaj
```

private_keys

E' anche possibile interrogare per le chiavi private e pubbliche di uno specifico wallet, ovviamente il wallet deve essere già aperto e bisogna reinserire di nuovo la password di blocco.

Positionals:

none

Options:

-n,--name *TEXT* - The name of the wallet to list keys from, otherwise - default
--password *TEXT* - The password returned by wallet create

Usage:

```
cleos wallet private_keys
```

Come ottenere EOS

EOS al contrario di altre crypto monete, ad esempio Bitcoin, non può essere minata dai semplici utenti e quindi è necessario investire e comprarli direttamente da chi li offre per esempio:

- tramite un Exchange, centralizzato o decentralizzato (DEX) che permette si a di comprare EOS tramite i mezzi di pagamento normali, cioè carta di credito o bonifico bancario o vendendo un asset per comprare EOS sempre sulla

piattaforma stessa;

- oppure riceverli in cambio di prestazioni di beni e servizi, quindi farsi pagare con quella crypto moneta dato che possiamo sfruttare i codici QR che possiamo generare direttamente dal wallet EOS e quindi mostrando solo quel codice il trasferimento sarà effettuato nel giro di pochi minuti e senza costi di commissioni, come avviene per altre blockchain tipo Bitcoin;

- giocando a varie di dApp, come quelle simili al casinò che permettono di scommettere e vincere EOS in base a quanto si è puntato, oppure altre dApp permettono di vendere degli asset del gioco tramite un apposito Marketplace

interno e quindi compare e vendere asset in cambio di EOS, oppure anche offrendo ricompense giornaliere per login o per giveaway temporanei;

- diventare un BP e quindi essere ricompensato con degli EOS ed in base alla posizione generale che si occupa la ricompensa è minore o maggiore.

Come vedere il prezzo di EOS

Come è consueto in questo mondo il valore e di conseguenza il prezzo di una crypto moneta lo decide il mercato che in base alla domanda e all'offerta viene determinato in quel momento e quindi ci possono essere momenti positivi o

negativi del valore stesso.

Per EOS il totale della supply è di circa superiore al miliardo di token attualmente (precisamente 1,017,807,409.7551) e di cui poco meno del 50% è messo in stake (cioè bloccato all'interno degli account per fornire potenza agli stessi) quindi di fatto quello che viene scambiato, cioè acquistato e venduto, rappresenta solo la metà di quello che potrebbe effettivamente rendere quindi si crea una specie di scarsità involontaria della crypto, anche perché se si togliesse dallo stake tutti i token allora nessuna dApp funzionerebbe e quindi risulterebbe inutile la vendita della crypto.

Alla data di oggi, 29-06-2019, ci

sono poco meno di 1,3 milioni di account diversi quindi in teoria almeno 1 milione di utenti effettivi, dato che un utente può avere e gestire più di un account EOS, e possiamo vedere che lentamente dalla sua creazione sempre più persone si sono avvicinate a questa blockchain.

Comunque ci sono diversi servizi dove è possibile vedere e controllare il prezzo e l'andamento della crypto moneta:

- CoinMarketCap, di sicuro quello più famoso che permette di avere numerosi dettagli relativi alle crypto e consente di vedere anche i principali Exchange e il loro andamento in termini del prezzo della crypto e dei volumi giornalieri per

farsi un'idea se ci sono persone che stanno facendo trading della stessa;

- OpenMarketCap, simile a quello precedente ma con dei parametri più stringenti e quindi utilizzando solo fonti fidati per la loro visione;

- diversi Exchange dove poter oltre controllare l'andamento del prezzo è possibile acquistare le crypto monete.

Come mandare e ricevere EOS

La procedura per trasferire EOS o token che si basano su questa blockchain, è abbastanza semplice e prevede l'utilizzo di un wallet sia esso desktop oppure mobile, quello che ci serve è l'account

di destinazione ed ovviamente l'importo che vogliamo trasferire.

La procedura è abbastanza standardizzata ed infatti una volta che ci siamo collegati con il nostro account non basta altro che andare nella sezione dedicata all'invio o al ricevimento ed inserire i suddetti parametri:

Transfer From Account (Linked to your Scatter)	Transfer To Account
Account to transfer from	Account to receive the transfer
Quantity	Token Symbol
0.0001	EFUK EFX EGT (EOS Game Token) EKD EMDS EMT (Emanate) ENB (ENB Platform) ENU EOS
Memo (optional)	
Transfer by EOSX	
TRANSFER	

L'importo minimo per il trasferimento è di 0,0001, indipendentemente dal tipo di token che scegliamo, inoltre è possibile

scrivere un memo (una nota che decidiamo noi di scrivere) questo è opzionale ma per la maggior parte degli Exchange è obbligatorio proprio per precisare il tipo di trasferimento che si effettua.

Se invece vogliamo ricevere un trasferimento allora dobbiamo fornire il nome del nostro account al destinatario oppure il codice QR che avremmo generato dalla nostra app o software, ed aspettare di ricevere la transazione, che avviene dopo pochi secondi e poi dopo pochi minuti diventa irreversibile

Capitolo 3

Come funziona la blockchain
di EOS

Per capire come funziona l'ecosistema di EOS dobbiamo partire dal suo protocollo di consenso ossia la DPoS (Delegated Proof of Stake), che è diversa da quella della PoW (Proof of Work) dove i minatori risolvono puzzle crittografici per minare i blocchi che andranno ad aggiungersi alla blockchain, consumando molta energia e potenza computazionale proprio per dimostrare il loro lavoro e dove una volta che verranno verificati e convalidati

riceveranno il relativo token, ed è diversa anche dalla PoS (Proof of Stake) dove il processo di mining viene virtualizzato ed i miner sono sostituiti dai validatori, dove questi ultimi bloccano o congelano parte dei loro token, che viene definito stake e dopo di che possono iniziare a validare i blocchi e quando scoprono un nuovo blocco che potrebbe essere aggiunto alla blockchain e per validarlo puntano una scommessa sul blocco e se questo viene allegato allora il validatore riceverà una ricompensa in proporzione di quello che ha scommesso.

Prima di tutto nell'ecosistema di EOS chiunque detiene dei token sulla blockchain di EOS può essere

selezionato come block producer tramite un sistema di votazione continua, quindi chiunque ha la possibilità di partecipare alla produzione dei blocchi in maniera proporzionale al totale dei voti che ha ricevuto rispetto agli altri block producer.

I blocchi sono prodotti in 21 round ed all'inizio di ogni round vengono scelti 21 block producer, i primi top 20 sono scelti in maniera automatica mentre il 21° è scelto in maniera proporzionale al numero dei voti rispetto a quelli degli altri block producer. I BP sono poi mescolati usando un numero pseudo casuale che deriva dal block time, permettendo di avere una connettività bilanciata tra tutti i block producer. Per

mantenere la produzione regolare dei blocchi, 126 blocchi per ogni round (6 per ognuno dei 21 BP), 1 ogni 0,5 secondi, il block time viene settato a 3 secondi e dove se non produce almeno un blocco ogni 24 ore il BP viene escluso e non viene preso in considerazione.

In un sistema DPoS non possiamo trovare il tradizionale fork, che potrebbe avvenire sulle altre blockchain con protocolli di consenso diversi, perché non c'è competizione per trovare i blocchi ma c'è la cooperazione tra i BP ed anche nel caso avvenisse una specie di fork il protocollo di consenso cambia automaticamente verso la catena più lunga.

Come vengono confermate le transazioni in un sistema DPoS

Una blockchain che utilizza il protocollo DPoS tipicamente ha il 100% della partecipazione dei block producer, ed una transazione è di solito confermata in 1,5 secondi dal momento in cui viene trasmessa, con una certezza del 99,9%, per avere l'assoluta certezza per la validità della transazione, bisogna solo attendere per 15 dei 21 BP (la maggioranza dei 2/3) che arrivino al consenso.

Cosa succede se un evento come un fork, causato per negligenza o malevolo,

accade? In questo caso tutti i nodi, di default, non passeranno al fork se questo non è finalizzato prima da 15/21 dei BP, indipendentemente dalla lunghezza della catena, infatti ogni blocco deve essere approvato dai 15/21 dei BP per far parte della catena.

Dato che la creazione dei blocchi avviene in poco tempo, è possibile avvisare i nodi se li trovano su una catena maggiore o minore nell'arco di 9 secondi, dove ricordiamo che tra un blocco e l'altro c'è un tempo di 3 secondi, quindi se un nodo perde 2 blocchi consecutivi c'è il 95% di probabilità che si trovano in un fork minore, mentre se un nodo perde 3 blocchi allora ci sarà la probabilità del

99% che si trovano sulla catena minore.

Una funzionalità di EOS, la TAPOS

La TAPOS (Transaction As Proof of Stake) è una caratteristica di EOS e dove ogni transazione nel sistema deve avere l'hash della testa del blocco recente ed in questo modo si previene transazioni ripetute su diverse catene e segnala alla rete ce un utente con il suo stake sono in un fork particolare.

Con questo sistema si eliminano le fee dalle transazioni dove nell'ecosistema di EOS lavora su un modello dove gli utenti hanno la proprietà delle risorse, proporzionalmente a quanto hanno

messo in stake, piuttosto che pagare per ogni transazione. In pratica se si hanno in stake X EOS allora potrai fare un numero di operazioni $X * n$ transazioni e così si elimina di fatto il costo delle transazioni.

Quindi basta mettere degli EOS in stake nel proprio account per avere una parte della potenza computazionale della e delle risorse di tutto l'ecosistema su cui gira EOS, in proporzione alle risorse che ha bloccato l'utente.

Le transazioni in EOS

Un blocco sulla blockchain di EOS può includere differenti transazioni

all'interno e dove ogni transazione può contenere una o più azioni differenti collegate tra loro, dove ogni interazione con la blockchain è definita transazione, quindi non solo movimenti di token da un account ad un altro, ed ogni azione è un'istruzione che performa in base all'azione data.

Block #63,487,282

Summary: Irreversible

Block Height:	63,487,282 ← Prev Next →
Timestamp:	Jun-14-2019, 07:05:41 PM CEST
Producer Name:	eossv12eossv
Block ID:	03c8bd32085a59c169e105953522a115655d2c7e4db67847bed8a8da86543f68

Additional Information

Schedule Version:	963
Number of Transactions:	31
Number of Actions:	33
Previous Block:	63,487,281
Next Block:	63,487,283

Transactions		Raw				
ID	expiration	CPU Usage	NET Usage	Number of Actions		
a48567dc23e8e54f12bd9cad0967a046b71ae58ab828f350673a7b56e1461417	N/A	100 μ s	0 Bytes	N/A		
3565721d81bb1dc394ec9979833e8830eedcea4236dff713acb4b46ad6ff67c	Jun-14-2019, 07:06:10 PM	276 μ s	384 Bytes	1		
11acc85fb34b08320c6842785fe7a7762504ba4ffac40997ffe5d6d95704d055	Jun-14-2019, 07:06:40 PM	247 μ s	194 Bytes	1		
f41aab49b1ec618d9c5f9e7c26ceab92e0ff11bfb56b2d7c06dc0432a6f1d156	Jun-14-2019, 07:06:10 PM	286 μ s	128 Bytes	1		
d2389e70e76c7328e87c95e2458b9825baac73252974cb3d23ea0c66bfa63ae	Jun-14-2019, 07:06:10 PM	235 μ s	296 Bytes	1		
6f7ac2599388374452300c038b3b0342c7b5c745c0bb88c76a32102c081a0e2e	Jun-14-2019, 07:06:09 PM	1,166 μ s	336 Bytes	3		
a5e597d573447c73b7579f6bc28400660602a0f418fce11233861f7d434f9e69	Jun-14-2019, 07:06:10 PM	189 μ s	264 Bytes	1		
140bdf15f777de26e28dba45f0a01cd05215ae3d7cd09b82f897cc464f45c51	Jun-14-2019, 07:06:38 PM	163 μ s	192 Bytes	1		
8425e9560e7dc92784cdf7b8b741975efa60159caa691b99a6a9c0ffa3902dd3	Jun-14-2019, 07:06:38 PM	165 μ s	192 Bytes	1		
918900f073eb583baa4c932e21a52dca08fd840026cf139514d563ae0422ad	Jun-14-2019, 07:06:38 PM	228 μ s	104 Bytes	1		
68b106bcbb22854649698f9bea877c2ed76af95614f2a531409917b4de5ed57a	Jun-14-2019, 07:08:40 PM	272 μ s	104 Bytes	1		
9ad8326b58e27f8f9bbaed5df81dd7f0d6bfc34d7a87dfd16f7ddc799facc377	Jun-14-2019, 07:06:38 PM	167 μ s	192 Bytes	1		

Alcuni tipi di transazioni su un account EOS

Ci sono diversi tipi di transazioni (52 in totale per il momento ma possono aumentare o diminuire a seconda degli aggiornamenti che ci saranno) che

possiamo trovare in un account EOS, alcune di esse appaiono perché è l'utente stesso che le ha avviate o qualcuno che abbia accesso alle chiavi di quell'account, mentre altre transazioni si trovano perché sono azioni create da altri, come ad esempio nel caso degli airdrop.

bidname	bidrefund	buyram	buyrambytes	buyrex	canceldelay	claimrewards
closerex	cmdrexorder	consolidate	defcpuloan	defnetloan	delegatebw	deleteauth
deposit	fundcpuloan	fundnetloan	init	linkauth	mvfrsavings	mvtsavings
newaccount	onblock	onerror	refund	regproducer	regproxy	rentcpu
rentnet	rexexec	rmvproducer	sellram	sellrex	setabi	setacctcpu
setacctnet	setacctram	setallimits	setcode	setparams	setpriv	setram
setramrate	setrex	undelegatebw	unlinkauth	unregprod	unstaketorex	updateauth
updaterex	updtrevision	voteproducer	withdraw			

**Transazione - New Account
(newaccount)**

Questa transazione, come dice il nome, avviene quando un nuovo account è stato creato, questa azione è iniziata con le chiavi dell'account che si possiede e quindi creato sapendo di questa creazione, inoltre ogni account creato sulla rete di EOS necessita di un poco di RAM per immagazzinare alcune informazioni come il nome dell'account, la chiave pubblica e la chiave privata insieme ad altre informazioni importanti collegati all'account stesso, quindi è normale vedere che alla transazione della creazione dell'account, venga accompagnata l'azione Buy RAM. Se si è avviata questa azione allora non c'è da preoccuparsi, ma è doveroso ricordare

che non si può cancellare un account e quindi quei token utilizzati per la creazione dell'account sono bloccati per sempre, quindi meglio non creare molti account se non necessario.

eosio

New Account

lumilovelumi active

lumilovelumi created the account kalengtempek with permissions

@owner

```
{
  "accounts": [],
  "keys": [
    {
      "key": "E0571Q5kFuyv1Nl0gwF9RkGat1VJPey79Tb44TS0pJT7HhEfkfZp2",
      "weight": 1
    }
  ],
  "threshold": 1,
  "waits": []
}
```

@active

```
{
  "accounts": [],
  "keys": [
    {
      "key": "E0571Q5kFuyv1Nl0gwF9RkGat1VJPey79Tb44TS0pJT7HhEfkfZp2",
      "weight": 1
    }
  ],
  "threshold": 1,
  "waits": []
}
```

Transazione - Buy RAM (buyram)

La RAM è un veloce accesso di

scrittura/lettura per salvare dati, serve per immagazzinare dati ed è necessario acquistarla, per la maggior parte degli account si può vedere questa transazione quando si crea un nuovo account sulla blockchain di EOS, comunque è anche possibile comprare RAM per conto proprio e c'è anche un mercato speculativo riguardo la RAM dato che segue un mercato dedicato, si utilizza l'algoritmo Bancor, ossia il prezzo della RAM è settato dal sistema EOSIO e automaticamente aggiusta il prezzo in alto o in basso in base alla supply e alla domanda, quindi il totale di EOS che si ha indietro quando si vende RAM non è uguale a quello che si era utilizzata per comprare lo stesso quantitativo di RAM,

inoltre una piccola fee (tassa) è applicata ad ogni acquisto e vendita della stessa dove finanzia REX (Resource Exchange).

Buy RAM	lumilovelumi bought 3,000 bytes of RAM for kalengtempek	
Transfer - eosio.token	lumilovelumi → eosio.ram 0.3095 EOS	Memo: buy ram
Transfer - eosio.token	lumilovelumi → eosio.ramfee 0.0016 EOS	Memo: ram fee
Transfer - eosio.token	eosio.ramfee → eosio.rex 0.0016 EOS	Memo: transfer from eosio.ramfee to eosio.rex Short Memo

Transazione - Buy RAM Bytes (buyrambytes)

Simile a quella precedente quindi segue gli stessi passaggi.

Transazione - Delegate Bandwidth (delegatebw)

Quando si parla di Bandwidth (larghezza della banda) bisogna distinguere la Network Bandwidth e la CPU Bandwidth. La Network Bandwidth è il tasso di trasferimento dei dati sul network di EOS mentre la CPU Bandwidth è come le transazioni girano, se è una transazione complicata, sarà consumata più CPU bandwidth.

L'azione di delegate bandwidth (o anche stake) specifica l'ammontare di CPU o Network bandwidth da allocare ad un account, e la procedura di allocazione è

immediata mentre quella per rimuovere (undelegate o unstake) necessita di 3 giorni, operazione che richiede ovviamente l'utilizzo delle proprie chiavi per questo tipo di operazione. Inoltre questa procedura può essere effettuata sia sul proprio account che ad un altro account EOS qualsiasi.

eosio - delegatebw

Stake CPU/NET

lumilovelumi staked 0.0001 EOS of CPU and 0.0001 EOS of NET for kalengtempek

Transfer - eosio.token

lumilovelumi → eosio.stake 0.0002 EOS

Memo: stake bandwidth

Transazioni - Undelegate Bandwidth (undelegatebw)

Questa è la procedura inversa rispetto alla delegate bandwidth, praticamente si

dice alla rete di EOS di fare l'unstake della NET o CPU bandwidth, poi una volta che l'operazione è rilasciata serviranno 3 giorni ($3 \cdot 24 \cdot 3600$) per completare il processo, quindi il modo migliore per trattenere per un lungo periodo i propri token EOS (HODL) è meglio metterli in stake e poi fare l'unstake 3 giorni prima che si hanno intenzione di venderli.

Operazione che richiede ovviamente l'utilizzo delle proprie chiavi per questo tipo di operazione. Inoltre questa procedura può essere effettuata sia sul proprio account che ad un altro account EOS a cui si è fatto lo stake.

eosio - undelegatebw

Unstake CPU/NET

monarch.x unstaked 0.1 EOS of CPU and 0.0001 EOS of NET from dcdddwatched

Transazione - Vote Producer (voteproducer)

Quest'azione essenzialmente avviene quando si esprime il proprio voto per uno o più BP (Block Producer), l'azione avviene immediatamente e la potenza del voto dipende da quanti EOS in stake ha l'account sia sul proprio account che su quello degli altri; è importante che si voti periodicamente, anche riconfermando quegli attuali, perché una volta che si vota inizia il periodo di

decadenza, quindi in un anno se non si vota il valore di quel voto sarà la metà di quanto era all'inizio, votare i BP è indispensabile per l'ecosistema di EOS proprio per garantire un sistema sempre efficiente, inoltre, per il momento, ogni account può votare fino a 30 BP così da distribuire il proprio voto su più progetti ed avere anche i BP minori una possibilità.

Ho scritto per il momento perché ci sono alcune proposte di referendum che vogliono cambiare questo sistema e portando 1 account 1 voto così da evitare dei possibili cartelli e quindi impedire una corretta rotazione dei BP ed evitare che controllino tutta rete.

Transazione - Claim Rewards (claimrewards)

Questa transazione la si vede negli account dei Block Producer, dove richiedono il premio periodicamente per aver prodotto i blocchi, vera fonte di guadagno dei Block Producer per mantenere e proteggere la rete, questa transazione ovviamente non la vediamo nei nostri account a meno che non si è un BP, l'operazione è manuale e non automatica e che utilizza la chiave assegnata al loro account, e di solito la si può vedere ad intervalli regolari.

Claim Rewards	cypherglass claimed block producer rewards		
Issue - eosio.token	eosio.token issued	552.3258 EOS	to eosio
Transfer - eosio.token	eosio → eosio.saving	441.8607 EOS	Memo: unallocated inflation
Transfer - eosio.token	eosio → eosio.bpay	27.6162 EOS	Memo: fund per-block bucket
Transfer - eosio.token	eosio → eosio.vpay	82.8489 EOS	Memo: fund per-vote bucket
Vote Pay	eosio.vpay → cypherglass	276.4384 EOS	Memo: producer vote pay

Transazione - Update Authentication (updateauth)

Tra le transazioni più importanti che può fare un account e dove si possono modificare le chiavi, sia la chiave di proprietà che la chiave attiva e dove bisogna stare attenti soprattutto alla

chiave di proprietà dato che con quella si può cambiare la chiave attiva e quindi in caso di account compromesso è possibile gestire il tutto.

Update Auth

sendrayeos11 set the permission owner to have the authentication

```
{
  "accounts": [],
  "keys": [
    {
      "key": "E0S6yDtG5tJ8U4qbwC61VvV2brAVGvspTvjZLG6bvZ5MXU94uQ2dg",
      "weight": 1
    }
  ],
  "threshold": 1,
  "waits": []
}
```

Update Auth

sendrayeos11 set the permission active with the parent permission owner to have the authentication

```
{
  "accounts": [],
  "keys": [
    {
      "key": "E0S6yDtG5tJ8U4qbwC61VvV2brAVGvspTvjZLG6bvZ5MXU94uQ2dg",
      "weight": 1
    }
  ],
  "threshold": 1,
  "waits": []
}
```

Transazione - Transfer (transfer)

Quest'azione si verifica quando i token

EOS vengono mossi sulla rete, e questo può significare sia che quei token sono arrivati al nostro account e sia che i token hanno lasciato il nostro account, dove per i ricevimenti di token non è richiesta nessuna chiave mentre per gli invii dei token è necessaria la chiave, inoltre ogni azione di trasferimento ha un memo allegato insieme, non sempre compilato dato che è opzionale scriverci dentro, dove di solito viene utilizzato per avere un dettaglio ulteriore del tipo di trasferimento.

eosio.token - transfer

Transfer - eosio.token

bnt2eoscnvrt → sendrayeos11 1.1638 EOS

Memo: convert

Transazione - Airdrop Action (airdrop)

Quest'azione si può vedere quando riceviamo un airdrop sul nostro account, infatti gli airdrop possono essere definiti come il processo dove un progetto crypto distribuisce i propri token ai wallet di tutti gli utenti o parte di essi, in modo completamente gratuito, e di solito non è richiesta nessuna azione da parte dell'utente e quindi poi l'utente può decidere di tenere o vendere.

emanateoneos

Issue - emanateoneos

airdropsdac1 airdrop

emanateoneos Issued 10.0000 EMT to alanpeterson

Come il wallet interagisce con la blockchain di EOS

Un wallet per operare sulla blockchain di EOS deve interagire con nodeos (node + eos, che è un demone, cioè un programma eseguito in background, senza che sia sotto il controllo diretto dell'utente ma che fornisce un servizio allo stesso), che è un programma che fa parte dell'ecosistema EOSIO, rappresenta il core node di EOSIO che può essere configurato con dei plugin per avviare un nodo e può essere usato ad esempio per la produzione di blocchi, API dedicate ed anche per sviluppo locale, quindi in pratica è la blockchain dove ci sono tutti i database,

i contratti in memoria e così via, dove il wallet, anche'esso un demone, lo possiamo assimilare ad un altro server, ed ascolta le altre porte.

Se vogliamo creare una transazione, la stessa è fatta da diverse azioni, che abbiamo visto in precedenza, e queste azioni sono delle operazioni, che vanno a formare la transazione, poi si prende questa transazione e chiediamo alla blockchain, nodeos, gli ultimi blocchi, perché se vogliamo marcare la transazione ci serve sapere quali sono gli ultimi blocchi che saranno aggiunti alla stessa per avere un blocco irreversibile.

Inoltre dobbiamo chiedere alla

blockchain quali chiavi abbiamo bisogno per firmare quella transazione affinché sia valida, quindi a quel punto viene fatta una richiesta al wallet dove viene mandata la transazione più le relative chiavi per firmare la transazione, poi il wallet prende la relativa transazione la firma, senza comunicare con l'esterno, poi restituisce la firma e la relativa transazione dove poi prendiamo questa firma ritorniamo sulla blockchain per la pubblicazione, successivamente spingiamo la transazione firmata e poi questa verrà propagata e tutti eseguiranno l'azione, aggiorneranno il loro stato interno della memoria e verificheranno la firma, se la firma è corretta allora la transazione

viene accettata, mentre se non è corretta la firma sarà rifiutata dal primo nodo così da impedirne la diffusione.

In questo modo abbiamo il wallet, che gestisce tutte le chiavi segrete che abbiamo, e nodeos che è di fatto pubblica e comunica con il wallet dove possiamo leggere e spingere le transazioni, dove l'unica operazione di scrittura delle operazioni sulla blockchain è quella di spingere le transazioni e quindi il database è aggiornato attraverso queste transazioni.

Cosa sono la RAM, CPU e NET

Come più volte accennato il sistema di

EOS si basa su 3 elementi fondamentali per funzionare: RAM, CPU e NET e sono tutti uno complementare all'altro e dove in base a quello che dovremmo fare saranno richieste più o meno risorse al nostro account e tutte queste risorse della rete sono fornite dai BP, inoltre la qualità dell'hardware che utilizzano gli stessi è fondamentale anche per gli utenti così possono sapere che tipo di hardware dispone un relativo BP e quindi decidere nel caso di votarlo.

Questo significa anche che i BP permettono alle dApp e agli utenti di utilizzare il proprio hardware per

salvare e processare le transazioni, permettendo in questo modo agli utenti di accedere a diversi server dei BP e risolvendo il problema della centralità del tutto.

RAM

In ambito dei computer la RAM (Random Access Memory) è una forma di salvataggio dati dove vengono salvate le istruzioni che si stanno usando in quel momento, e per questo il suo compito è quello di mantenere quelle istruzioni per poco tempo e permette velocità di lettura e scrittura abbastanza rapide. Mentre invece la RAM in EOS è

leggermente diversa perché non corrisponde esattamente a quel concetto dove tutte le risorse, eccetto la CPU e la NET, possiamo, in maniera grossolana, identificare come RAM ed è come se fosse una specie di database per salvare informazioni in modo permanente come ad esempio le chiavi dell'account o il relativo saldo.

La RAM è molto importante per gli sviluppatori delle dApp infatti tutte le operazioni che servono per la dApp sono salvate nella RAM e dove se questa non sia sufficiente, allora alcune operazioni non potranno essere effettuate e lo smart contract relativo non potrà essere eseguito.

Dato che la RAM va comprata a

parte allora questo segue un mercato ed andamento distaccato rispetto all'andamento del token EOS, perché quando si libera RAM non è detto che si ha lo stesso importo in EOS che si era utilizzato per comprarla e quindi questo è quello che viene definito trading della RAM, dove non c'è un scambio ed un trasferimento peer-to-peer ma la controparte di tutti i partecipanti è il system market maker, in questo caso uno smart contract.

Il sistema attuale, passando dal modello precedente in Dawn 3.0 a quello attuale Dawn 4.0, utilizza l'algoritmo Bancor, dove non setta il prezzo della RAM ma

offre all'acquirente ed al venditore un tasso precedente che si era stabilito nel mercato e quindi il tasso sarà diverso da quello che offrirà l'algoritmo e quindi i trader cercheranno di fare i loro scambi a un prezzo vicino a quello di mercato, dove il vantaggio è che il mercato a settare il prezzo e questo sistema risponde semplicemente a quello che offre il mercato.

Inoltre quando si crea un nuovo account un minimo di RAM è richiesta per il tutto e quindi bisogna comprarla se non se ne dispone a priori, dove al momento circa 4000 byte dovrebbero bastare per creare un account e possiamo tranquillamente affermare che, praticamente, bisogna pagare per avere

un nuovo account.

Va ricordato anche che quando un account EOS accetta un nuovo tipo di EOS token per la prima volta, questo viene registrato nella blockchain e quindi necessita di spazio per registrare questa informazione così poi ogni transazione ulteriore che coinvolge l'account con quel tipo di token non costerà più niente.

Come comprare e vendere RAM

L'acquisto e la vendita della RAM per un account EOS, o il nostro o quelle di qualcun altro, è abbastanza semplice e si può effettuare tramite una applicazione

su smartphone, che gestisce il wallet EOS e che permette una gestione completa del proprio account, oppure tramite applicazioni desktop come Scatter oppure tramite i vari servizi online di block explorer che permettono una gestione avanzata del proprio account. Quindi una volta che ci troviamo sull'applicazione in questione dobbiamo per prima cosa avere degli EOS da investire nell'acquisto della RAM e se non ne abbiamo allora li dovremmo recuperare prima in qualche modo, anche comprandoli da qualche parte, poi ci rechiamo nella sezione dedicata alla gestione dell'account oppure alla voce buy/sell RAM e da lì comprare o vendere la RAM di cui

abbiamo bisogno dove possiamo scegliere se comprare in EOS o in Byte a seconda di come ci troviamo meglio a fare i calcoli.



List of Tools

RAM

Buy RAM

Sell RAM

Buy RAM

RAM Receiver:

Account receiving RAM (may be same as payer to buy for yourself)

Buy in EOS or Bytes?

EOS Bytes

Amount of RAM to Buy in EOS

Amount of RAM to Buy in EOS

= 0 Bytes

[Buy RAM](#)

List of Tools

RAM

Buy RAM

Sell RAM

Sell RAM

Bought: 106267 Bytes = 10.0773 EOS

Used: 14529 Bytes = 1.3778 EOS

Free (Can sell): 91738 Bytes = 8.6995 EOS

Amount of RAM to Sell (Bytes)

25% 50% 75% 100%

Selling 0 Bytes for 0 EOS

[Sell RAM](#)

Come comprare e vendere RAM con Cleos

system buyram

Permette di comprare la RAM.

Positionals:

payer *TEXT* - The account paying for RAM

receiver *TEXT* - The account receiving bought RAM

tokens *TEXT* - The amount of EOS to pay for RAM

Options:

-h,--help Print this help message and exit

-x,--expiration *TEXT* - set the time in seconds before a transaction expires, defaults to 30s

-f,--force-unique - force the transaction to be unique. this will consume extra bandwidth and remove any protections against accidentally

issuing the same transaction multiple times

`-s,--skip-sign` Specify if unlocked wallet keys should be used to sign transaction

`-d,--dont-broadcast` - Don't broadcast transaction to the network (just print to stdout)

`-r,--ref-block TEXT` set the reference block num or block id used for TAPOS (Transaction as Proof-of-Stake)

`-p,--permission TEXT` - An account and permission level to authorize, as in 'account@permission' (defaults to 'account@active')

`--max-cpu-usage-ms UINT` - set

an upper limit on the milliseconds of cpu usage budget, for the execution of the transaction (defaults to 0 which means no limit)

`--max-net-usage UINT` - set an upper limit on the net usage budget, in bytes, for the transaction (defaults to 0 which means no limit)

Usage:

```
#buy ram for yourself
```

```
cleos system buyram someaccount1  
someaccount1 "10 EOS"
```

```
#buy ram for someone else
```

```
cleos system buyram someaccount1  
someaccount2 "1 EOS"
```

system sellram

Permette di vendere la RAM

Positionals:

account *TEXT* - The account to receive EOS for sold RAM

bytes *UINT* - Number of RAM bytes to sell

Options:

-h,--help Print this help message and exit

-x,--expiration *TEXT* - set the time in seconds before a transaction

expires, defaults to 30s

-f,--force-unique - force the transaction to be unique. this will consume extra bandwidth and remove any protections against accidentally issuing the same transaction multiple times

-s,--skip-sign Specify if unlocked wallet keys should be used to sign transaction

-d,--dont-broadcast - Don't broadcast transaction to the network (just print to stdout)

-r,--ref-block *TEXT* set the reference block num or block id used for TAPOS (Transaction as Proof-of-Stake)

-p,--permission *TEXT* - An account and permission level to authorize, as in 'account@permission' (defaults to 'account@active')

--max-cpu-usage-ms *UINT* - set an upper limit on the milliseconds of cpu usage budget, for the execution of the transaction (defaults to 0 which means no limit)

--max-net-usage *UINT* - set an upper limit on the net usage budget, in bytes, for the transaction (defaults to 0 which means no limit)

Usage:

```
#sell 1000 bytes
```

```
cleos system sellram someaccount1
```

1000

CPU

Ogni volta che si invia una transazione sulla rete di EOS i BP devono processare quella transazione, e la CPU è la lunghezza del tempo, misurata in microsecondi (μs), che un BP deve spendere per processare e validare la transazione.

Dato che ogni BP ha un'infrastruttura diversa, la CPU è diversa per ogni BP e quindi questo ha portato allo sviluppo di strumenti per misurare, in modo più

preciso possibile, la relativa velocità di esecuzione per ogni BP^[8] .

Nell'ecosistema di EOS i blocchi sono creati ogni 500 millisecondi e per aiutare i BP ad avere abbastanza tempo per diffondere questi blocchi nel mondo c'è un tempo di processo per blocco di 200 millisecondi entro cui il BP deve validare il blocco prima di trasmetterlo nella rete e quindi questo permette di avere 300 millisecondi per propagare il tutto sulla rete stessa.

Un'altra costrizione è che entro i 200 millisecondi c'è anche una percentuale di soglia a quale tasso inizia il limite, infatti prima che questo limite venga raggiunto, tutti gli utenti possono fare transazioni liberamente sulla rete dato

che questa non si trova in "congestion mode" e dove si sorpassa questo limite, gli utenti saranno costretti ad utilizzare le proprie risorse messe in stake per la CPU.

Quando controlliamo il totale del tempo disponibile della CPU sul nostro account tramite un block explorer troviamo 2 parametri la CPU utilizzata e la CPU totale e se ricarichiamo la pagina possiamo vedere che il tempo cambierà a seconda della congestione della rete; la cosa positiva del tutto è che quando si consuma CPU poi questa si riprende dopo un periodo di 24 ore permettendo di fare ulteriori operazioni oppure basta mettere in stake degli altri EOS nella CPU per poter incrementare

il tempo di operatività.

Come per la RAM e per tutte le risorse dell'ecosistema EOS possiamo anche mettere in stake ad un account diverso dal nostro e quindi fornirgli la potenza necessaria per poter operare ed inoltre ci sono alcuni servizi che permettono di avere della CPU in maniera gratuita per poche ore per alcune dApp oppure servizi che permettono di prestare i propri token per

avere un interesse sul prestito e dove poi questi verranno richiesti da chi ha bisogno di token EOS.

Per quanto riguarda quanto conviene mettere in stake per poter operare con un account EOS senza avere problemi, questo dipende molto dall'utilizzo che se ne fa con un account dato che è molto variabile come parametro ma comunque si può fare una stima di operazioni^[9] che il proprio account può eseguire, dato che non tutte le dApp richiedono lo stesso quantitativo di EOS per operare, per esempio una dApp da gaming EOS Knight richiede di avere CPU in stake almeno 15 EOS, quindi una cifra importante per poter giocare.

Anche in questo caso per poter

mettere in stake o togliere dallo stake gli EOS bisogna sempre avvalersi di uno strumento che permetta la gestione di questo tipo di operazione sia esso un software per computer come Scatter o una applicazione mobile per smartphone.

List of Tools

Stake/Unstake

Stake CPU/NET

Unstake CPU/NET

Manual Refund

Stake CPU/NET

Receiver of Stake:

Amount of CPU to Stake (in EOS)

Amount of NET to Stake (in EOS)

Transfer stake to new account

List of Tools

Stake/Unstake

Stake CPU/NET

Unstake CPU/NET

Manual Refund

Unstake CPU/NET

Account name of who currently holds stake:

Amount of CPU to Unstake (in EOS)

Amount of NET to Unstake (in EOS)

Unstake

NET

Per quanto riguarda la NET (network), che come la RAM si misura in byte,

questa è un'altra risorsa a disposizione dell'account e che è la capacità di throughput sulla blockchain di EOS, quindi è il totale dei dati che possono essere mandati da un punto ad un altro in uno specifico lasso di tempo, o possiamo paragonare al traffico dati disponibile sulle nostre sim dove una volta superato quel limite ci fermiamo e per andare oltre dobbiamo pagare dove in questo caso nella blockchain di EOS significa che dobbiamo mettere in stake più token EOS alla risorsa NET, e di solito non è richiesta molta potenza per la stessa ma come al solito dipende da cosa anche perché un'azione consuma in media circa 150 byte e con 0.5 EOS (circa 487 Kb) è possibile effettuare

oltre 999 operazioni quindi sotto quel punto di vista è possibile mettere in stake un quantitativo di EOS inferiore rispetto a quanto necessario dalla CPU.

Anche in questo caso per poter mettere in stake o togliere dallo stake gli EOS bisogna sempre avvalersi di uno strumento che permetta la gestione di questo tipo di operazione sia esso un software per computer come Scatter o una applicazione mobile per smartphone.

List of Tools

Stake/Unstake

Stake CPU/NET

Unstake CPU/NET

Manual Refund

Stake CPU/NET

Receiver of Stake:

Amount of CPU to Stake (in EOS)

Amount of NET to Stake (in EOS)

Transfer stake to new account

List of Tools

Stake/Unstake

Stake CPU/NET

Unstake CPU/NET

Manual Refund

Unstake CPU/NET

Account name of who currently holds stake:

Amount of CPU to Unstake (in EOS)

Amount of NET to Unstake (in EOS)

Unstake

Come si mette in stake CPU e NET con Cleos

system delegatebw

Positionals:

from *TEXT* - The account delegating bandwidth

receiver *TEXT* - The account to delegate bandwidth from

stake_net_quantity *TEXT* - The amount of EOS to delegate for network bandwidth

stake_cpu_quantity *TEXT* - The amount of EOS to delegate for CPU bandwidth

Options:

-h,--help Print this help message and exit

-x,--expiration *TEXT* - set the time in seconds before a transaction

expires, defaults to 30s

-f,--force-unique - force the transaction to be unique. this will consume extra bandwidth and remove any protections against accidentally issuing the same transaction multiple times

-s,--skip-sign Specify if unlocked wallet keys should be used to sign transaction

-d,--dont-broadcast - Don't broadcast transaction to the network (just print to stdout)

-r,--ref-block *TEXT* set the reference block num or block id used for TAPOS (Transaction as Proof-of-Stake)

-p,--permission *TEXT* - An account and permission level to authorize, as in 'account@permission' (defaults to 'account@active')

--max-cpu-usage-ms *UINT* - set an upper limit on the milliseconds of cpu usage budget, for the execution of the transaction (defaults to 0 which means no limit)

--max-net-usage *UINT* - set an upper limit on the net usage budget, in bytes, for the transaction (defaults to 0 which means no limit)

Usage:

```
cleos system delegatebw accountname1  
accountname2 "1 SYS" "1 SYS"
```

Come togliere dallo stake CPU e NET con Cleos

system undelegatebw

Positionals:

from *TEXT* - The account undelegating bandwidth

receiver *TEXT* - The account to undelegate bandwidth from

unstake_net_quantity *TEXT* - The amount of EOS to undelegate for network bandwidth

unstake_cpu_quantity *TEXT* - The amount of EOS to undelegate for

CPU bandwidth

Options:

-h,--help Print this help message and exit

-x,--expiration *TEXT* - set the time in seconds before a transaction expires, defaults to 30s

-f,--force-unique - force the transaction to be unique. this will consume extra bandwidth and remove any protections against accidentally issuing the same transaction multiple

times

`-s,--skip-sign` Specify if unlocked wallet keys should be used to sign transaction

`-d,--dont-broadcast` - Don't broadcast transaction to the network (just print to stdout)

`-r,--ref-block TEXT` set the reference block num or block id used for TAPOS (Transaction as Proof-of-Stake)

`-p,--permission TEXT` - An account and permission level to authorize, as in 'account@permission' (defaults to 'account@active')

`--max-cpu-usage-ms UINT` - set an upper limit on the milliseconds of

cpu usage budget, for the execution of the transaction (defaults to 0 which means no limit)

`--max-net-usage UINT` - set an upper limit on the net usage budget, in bytes, for the transaction (defaults to 0 which means no limit)

Usage:

```
cleos convert pack_transaction '{
  "expiration": "2018-08-02T20:24:36",
  "ref_block_num": 14207,
  "ref_block_prefix": 1438248607,
  "max_net_usage_words": 0,
  "max_cpu_usage_ms": 0,
  "delay_sec": 0,
  "context_free_actions": [],
```

```
"actions": [{
  "account": "eosio",
  "name": "newaccount",
  "authorization": [{
    "actor": "eosio",
    "permission": "active"
  }]
},
  "data":
"0000000000ea305500a6823403ea3055
}
],
"transaction_extensions": []
}'
```

Come vedere il totale in stake con Cleos

E' possibile vedere una lista del totale messo in stake da un account, sia in formato testo che formato json.

system listbw

Positionals:

account *TEXT* - The account delegated bandwidth

Options:

-h,--help - Print this help message and exit

-j,--json - Output in JSON format

Usage:

```
cleos system listbw someaccount1
```

I Block Producer (BP)

Un Block Producer (BP) lavora per facilitare la rete di EOS e farla crescere in modo continuativo, inoltre i token holder possono votare per uno o più BP (massimo 30 attualmente), voto fondamentale e che deve sempre essere costante e che permette al BP di poter creare i nuovi blocchi sulla blockchain di EOS.

I migliori BP candidati sono classificati in base al numero di voti che

hanno ricevuto e saranno divisi in 2 gruppi distinti, i top 21 BP creeranno i nuovi blocchi, mentre i restanti candidati resteranno in sospenso (Standby Producer), e nell'attesa che ricevano abbastanza voti per entrare nella top 21, potranno aiutare comunità attraverso il loro full node, rendere la rete più sicura e decentralizzata, ma dovranno avere sia il potere computazionale che il throughput necessario per diventare top 21 e quindi iniziare il tutto in un istante.

Un aspetto da ricordare è quello che i BP non mettono in stake i loro token ma piuttosto mettono i loro investimenti nell'infrastruttura, nei membri del team, nella loro reputazione, così come nel loro futuro guadagno potenziale, infatti

grazie all' algoritmo aBFT (asynchronous Byzantine Fault Tolerance) i BP possono firmare ogni blocco che creano.

Una volta che 15 dei 21 BP ha firmato, e nessuna delle sue firme include un timestamp o nell'altezza del blocco, allora il blocco diventa irreversibile, dove attraverso questo metodo un BP può essere scoperto in maniera veloce se ha commesso errori o se ha un comportamento sospetto.

Per i BP l'unica forma di incentivo è dato dai token EOS e dove i BP saranno ricompensati tramite il sistema di inflazione dove quando vengono creati dei nuovi blocchi dei nuovi token EOS

vengono creati e distribuiti.

L'inflazione è del 5% e solo l'1% di questa va ai BP come premio, mentre il restante 4% va verso l'account eosio.saving per un WPS (Worker Proposals), e quindi i BP tramite questi premi possono investire nella rete, permettendo di crescere e di fornire nuovi servizi e strumenti utili per la comunità. di questo 1% viene diviso tra i top 21 BP dove viene dato lo 0,25% mentre il restante 0,75% viene dato agli SP (Standby Producer), che hanno ricevuto abbastanza voti per essere pagati 100 EOS per un giorno.

C'è stata una proposta fatta da eosio.meetone il 12 maggio 2019 e che se approvata dai BP, riduceva il tasso di

inflazione all'1% e dove si ferma l'invio dei token EOS verso l'account eosio.saving anche perché dopo un anno di attività della blockchain non c'è stata una concreta idea su come utilizzare quei fondi, e dove si è cambiato il parametro di sistema `continuous_rate` a 0.00995.

Ruolo del Block Producer

Come abbiamo detto in precedenza EOS utilizza un meccanismo di consenso che si chiama DPoS (Delegated Proof of Stake) per avere un accordo sui blocchi, dove le transazioni saranno applicate e poi alla fine saranno salvate all'interno

della blockchain.

Prima di tutto per diventare BP, qualsiasi entità si può proporre per produrre blocchi ed inizia una campagna di elezione, come in un Paese democratico, dove chiedono ai possessori di EOS (EOS holder) di votare per loro usando il peso dei loro token. Questo processo è continuativo ed è una transazione come le altre, dove ogni 126 secondi (chiamato round) i nodi della blockchain di EOS ricalcolano i BP in base ai voti accumulati fino a quel punto, poi i top 21 BP sono selezionati per produrre i blocchi, in un determinato ordine. Sono ricompensati per fare questo, come i miner di Bitcoin, ma invece di essere

sconosciuti come in quel caso, sono noti anche perché mettono in gioco la loro reputazione, dove la comunità può far sentire la propria voce per circa 700 volte al giorno, tramite questo processo.

Doveroso fare un breve accenno su come aBFT (asynchronous Byzantine Fault Tolerance) aiuta gli utenti di EOS. Raggiungere uno stato irreversibile è molto d'aiuto perché tutti possiamo fidarci e appoggiarsi sulle informazioni che ci sono sulla blockchain e permette alle dApp e alle aziende di affidarsi a quella informazione. Un semplice esempio di aBFT e di come viene usato nell'ecosistema di EOS è che il consenso sia raggiunto da almeno $2/3 + 1$ su ogni blocco creato, dove tutti i BP

hanno l'abilità di firmare i blocchi, quindi una volta che 15 BP hanno firmato il blocco è considerato irreversibile, cosa che avviene in pochi secondi.

Come abbiamo detto ogni blocco consiste in una testa (header) ed una lista di transazioni, che queste non sono solo transazioni di token ma qualsiasi cosa necessiti di un'esecuzione come ad esempio un voto o una dApp che registra un'informazione, mentre l'header include informazioni del blocco precedente, il timestamp, il merkle root (un albero crittografico delle transazioni) ed ovviamente il nome del BP che ha fatto quel blocco, inoltre se per caso un BP fallisce nella produzione del blocco

(dove è prevista dal whitepaper una punizione per il BP), per un motivo qualsiasi, le transazioni vengono prese dal prossimo BP nella linea ed il processo continua

Block #64,673,375

Summary:	Irreversible	Additional Information	
Block Height:	64,673,375 — Prev Next —	Schedule Version:	984
Timestamp:	Jun-21-2019, 03:54:50 PM CEST	Number of Transactions:	24
Producer Name:	jedaaaaaaaaa	Number of Actions:	31
Block ID:	03dad65fe3cce61567f05d6e416ca359b691c758b7853d849f63327432697ada	Previous Block:	64,673,374
		Next Block:	64,673,376

Una caratteristica che EOS offre è quella di permettere ai BP di rivisitare una precedente transazione se ci dovesse essere bisogno, cosa molto rara e che viene trattata con cura e quindi si prende sempre molto sul serio quando bisogna interagire in una decisione ed anche in

questo caso è necessario la votazione di almeno 15 BP. In passato esisteva una struttura, ECAF (EOS COre Arbitration Forum), che aveva la funzione di arbitro in diverse situazioni ed anche in casi relativi ad account e problematiche agli stessi come nel caso di truffe o danni e quindi, tramite una procedura dedicata riuscire a recuperare il maltolto e di congelare i fondi rubati, e questo forum fu istituito di default nella costituzione di EOS al vecchio articolo IX, dove su una controversia i BP producer prendevano una decisione su se e come intervenire, dato che loro hanno il controllo della rete e quindi possono modificare alcune situazioni spiacevoli, cosa che adesso non c'è più essendo

stato abolito il 12 aprile 2019:

 Transaction:
b330e224e7a58caa3ea534f5dcda5729813a107b4066c7cd9d1caeb584501229

Summary		Additional	
Block Number :	52,570,382	CPU Usage :	2,265 μ s
Block Time :	Apr-12-2019, 01:56:21 PM CEST	Net Usage :	0 Bytes
Status	Executed Irreversible	# of Actions/Traces :	1 / 1

Actions

Actions (1) Traces (1) RAM Deltas Raw

Contract	Name	Authorization	Data
eosio	eosio - setabi	eosio active	The ABI was updated Show ABI

Questa possibilità permette anche di sistemare e riscrive un codice malevolo per la blockchain oppure uno smart contract che opera in modo poco chiaro, allora anche in questo caso i BP potranno intervenire e sistemare il tutto in maniera efficace e veloce chiedendo come sempre il voto di almeno 15 BP

per attuare la modifica necessaria.

I BP dovranno anche fornire uno spazio di archiviazione adeguato per la rete, non solo per archiviare i blocchi già creati ma anche per dApp che gireranno sulla rete, infatti se una dApp permette di condividere un file multimediale, questo sarebbe gestito dal BP e quindi lo spazio aumenta sempre in continuazione per scalare la rete e per risolvere questo problema ci si affida al protocollo IPSF (InterPlanetary File System) che permette di avere una protocollo peer-to-peer dei file e velocizzare il tutto e rendere distribuita anche l'informazione.

Anche in materia di sicurezza, per quanto riguarda la custodia della rete

dato che sono i BP che gestiscono la rete e quindi proteggere la stessa da male intenzionati, che in questo mondo non mancano, quindi oltre a mettere in sicurezza tutto il loro hardware devono anche mettere in sicurezza le loro chiavi per firmare i blocchi dato che se queste andrebbero perse allora si avrebbe un controllo incredibile sulla rete e di creare danni potenzialmente devastanti, inoltre alcuni BP hanno il loro hardware decentralizzato su servizi di hosting così da non rischiare che si conosca l'ubicazione del server e poter fare danni, diretti o indiretti come una catastrofe naturale, anche fisici alle strumentazioni.

Infine, anche se non è richiesto da

niente e nessuno, il compito dei BP così come dei SP, è quello di informare e formare gli utenti e la comunità sui vari aspetti sia tecnici che pratici sull'ecosistema EOS, visto che comunque sono gli stessi utenti che votano per loro e quindi si creerebbe una specie di ritorno formativo per tutti coloro che volessero partecipare in modo attivo all'evoluzione della blockchain, ma anche interagire con altri sviluppatori anche dei vari progetti e dApp che ci sono o che ci saranno in futuro e quindi fornire il supporto necessario affinché questi possano operare al meglio della rete EOS.

Come si vota per un BP

Dopo aver fatto un poco di chiarezza sui BP vediamo adesso come effettivamente un utente può votare un BP.

Quando si vota si sta praticamente eleggendo il BP ad operare sulla rete in base alle nostre scelte, nel senso che noi crediamo che un determinato BP abbia le caratteristiche necessarie per mandare avanti il tutto, ovviamente possiamo anche votare o quelli che già sono tra i top 21 oppure altri SP che potrebbero entrare nella top se raggiungono un determinato numero di voti, quindi i top 21 BP non sono fissi e possono cambiare in qualsiasi momento.

In EOS è possibile votare in

qualsiasi momento ed essere cambiato in qualsiasi momento, i voti sono contati ogni 126 secondi, che è il tempo, round, per la creazione del blocco, dove ogni round ogni BP produce 12 blocchi ed ogni blocco è prodotto in 0,5 secondi, questo significa che l'elezione dei BP avviene ogni 2 minuti e 6 secondi.

Ogni account può votare per massimo 30 diversi candidati ed ognuno di questi che si è scelto riceverà i voti in maniera eguale in base ai token EOS che sono messi in stake nell'account, quindi se un account ha 10 EOS in stake e vota per 10 BP, ognuno di essi avrà un valore di 10 EOS su cui contare, quindi non si diluisce la potenza del voto tra tutti i

votanti, inoltre che i token devono essere messi in stake mentre quelli liquidi non rientrano nel conteggio.

Per votare è possibile usare diversi strumenti, come ad esempio un block explorer oppure una applicazione che permette di gestire il voto dalla stessa, tramite la voce Vote, dove per prima cosa bisogna collegarsi con il proprio account e si selezionano prima i BP e poi si clicca sul pulsante Vote, dopo di che bisognerà confermare l'operazione.

Selected Producers (11 / 30): big.one x starteosiop x eoshuobpool x eoslaomaocom x hoo.com x newdex.bp x zbeosbp11111 x eosflytomars x
 eossv12eossv x atticabeosb x eosinfstones x

VOTE

Select	Rank	BP Name	Status	Location	Links	Votes %	Total Votes	Rewards Per Day
<input checked="" type="checkbox"/>	1	big.one	Top 21	China	← 🌐 🐦	2.007 %	152,351,977 -140,148	771.6392
<input checked="" type="checkbox"/>	2	starteosiop	Top 21	China	← 🌐 🐦	1.989 %	150,990,361 -241,973	767.6423
<input checked="" type="checkbox"/>	3	eoshuobpool	Top 21	China	← 🌐 🐦	1.982 %	150,456,721 -181,886	766.0758
<input checked="" type="checkbox"/>	4	eoslaomaocom	Top 21	Japan	← 🌐 🐦	1.98 %	150,309,515 -6,802	765.6437
<input checked="" type="checkbox"/>	5	hoo.com	Top 21	Singapore	← 🌐 🐦	1.925 %	146,159,687 -124,058	753.4621
<input checked="" type="checkbox"/>	6	newdex.bp	Top 21	Cayman Islands	🌐 🐦	1.907 %	144,804,042 -145,157	749.4827
<input checked="" type="checkbox"/>	7	zbeosbp11111	Top 21	China	← 🌐 🐦	1.907 %	144,803,517 -101,328	749.4811
<input checked="" type="checkbox"/>	8	eosflytomars	Top 21	China	🌐	1.829 %	138,856,006 -342,884	732.0225

Dopo che avremmo confermato il tutto vedremo nel nostro account l'azione della votazione ed un ricapitolo dei BP che si sono votati:

cryptonauta1 voted for cypherglass, eoscanadacom, teamgreymass

C'è anche la possibilità di proxare il proprio voto ossia conferire il proprio

voto ad un proxy voter che voterà per noi, che praticamente è un account dove ha la potenza di tutti gli account che hanno delegato il loro voto, delega che può essere tolta in qualsiasi momento e che non impedisce all'account di votare ed in questo caso il voto dell'account riscrive quello del proxy, quindi se per caso il proxy ha votato per una proposta di referendum in modo positivo ma noi abbiamo un parere opposto allora in quel caso il voto dell'account andrà a sovrascrivere quello del proxy.

Vote Producers

cryptonauta1 proxied their vote to atlantemundo

Voting Accounts

Voter	Staked	Vote	Proxy
collintrcrypto	4,037,660.2518 EOS	Yes	Yes
investingwad	3,551,588.4258 EOS	Yes	Yes
lukeeosproxy	2,880,367.9990 EOS	Yes	Yes
ge3dknjugige	2,791,047.2652 EOS	Yes	-
ggq2tknagenes	1,442,911.2236 EOS	Yes	-
freedomproxy	668,269.9664 EOS	Yes	Yes
tokenika4dev	523,682.0191 EOS	Yes	Yes
eosnetworkoxx	168,257.5626 EOS	No	Yes
goodguys4eos	164,314.0816 EOS	Yes	Yes
gmytomjvhage	150,093.4880 EOS	Yes	-







Anche in questo caso si può utilizzare il proprio metodo e strumento preferito per votare il proprio proxy, in questo caso si può votare solo un proxy alla volta.

Filter Proxies: Registered All

Please log in to proxy.

bloksloproxy

SET bloksloproxy AS PROXY

Select	Rank	Name	Account	Proxied Accounts	Voting For	Total Proxy EOS
<input checked="" type="radio"/> Proxy	135	 Blok.io	bloksloproxy	233	23	418,300
<input type="radio"/> Proxy	1	 The Starkness	madeofstarks	158	27	8,623,102
<input type="radio"/> Proxy	2	 Colin Talks Crypto	collintcrypto	1326	30	7,057,503
<input type="radio"/> Proxy	3	 Infinity Stones Proxy	infstonespxy	11	17	5,667,215
<input type="radio"/> Proxy	5	 Brock Pierce Proxy	brockpierce1	413	30	4,866,590
<input type="radio"/> Proxy	10	 Newdex	newdexproxy1	569	27	3,916,650

Qualsiasi account può diventare un proxy e ci sono due tipi di proxy quelli normali e quelli registrati che hanno ulteriori informazioni, per registrare queste informazioni si può utilizzare qualche strumento in rete che permette questa funzione, poi si inseriscono tutti i parametri relativi, obbligatori, per avere un quadro chiaro e preciso sulla propria visione di voto, che altri utenti

potrebbero condividere oppure no.

VOTE VOTE VOTE BY PROXY REGISTER PROXY UNREGISTER PROXY

Allows you to vote on behalf of others.

Your Account Name (Linked to Scatter)

BECOME PROXY

SUBMIT PROXY INFO

Submit more info about your proxy account. See the [instructions](#) here.

Your Account Name (Linked to Scatter)

Full Name

The full name of your proxy

Logo

An URL to an image with the size of 256 x 256 px.

Slogan

A short description of your proxy intended to be shown in listings.

Philosophy

Description of proxy's voting philosophy.

Background

Background information / who is the proxy?

Website

An URL to a website, Reddit post, etc. with more information about your proxy.

Contact

Telegram	Account name	Steemit	Account name	Twitter	Account name	WeChat	Account name
-----------------	--------------	----------------	--------------	----------------	--------------	---------------	--------------

SUBMIT PROXY INFO

La questione del voto porta anche alla questione che potrebbero comprarsi i

voti i BP e quindi potrebbero agire in malo modo, cosa che è vietata dalla costituzione di EOS e che qualsiasi tentativo e pratica che faciliti questo sarà punito severamente dato che si deve mantenere il voto neutrale e sincero dell'utente finale così da avere un sistema più chiaro e trasparente possibile senza pressioni esterne o incentivi da parte dei BP che così avrebbero un vantaggio anche sugli altri BP onesti.

Affinché ci sia sempre del movimento sui voti ed indurre gli utenti a votare si è introdotto un sistema di decadenza annuale del voto che ne riduce la potenza stessa. La decadenza inizia una settimana dopo che si è votato

e se l'utente non rivota dopo questa settimana, la potenza del suo voto decade e se in un anno non ha rivotato allora la sua potenza si dimezza del 50%:


```
105 double stake2vote( int64_t staked ) {
106     /// TODO subtract 2000 brings the large numbers closer to this decade
107     double weight = int64_t( now() - (block_timestamp::block_timestamp_epoch / 1000)) / (seconds_per_day * 7) ) / double( 52 );
108     return double(staked) * std::pow( 2, weight );
```

Per i più tecnici possiamo vedere che tecnicamente il voto non decade ma i nuovi voti hanno un peso maggiore di quelli vecchi dove la voce `block_timestamp_epoch`, il software EOS.IO usa i secondi a partire dal primo gennaio 2000, mentre lo standard Unix timestamp conta i secondi a partire dal 1970, quindi se un utente vorrebbe mantenere alto il suo voto allora

dovrebbe farlo ogni settimana partendo dalle 00:00:01 UTC del sabato, inoltre la forza del voto è considerata piena se è stata performata un'azione di stake o unstake. Rivotando ogni settimana l'utente incrementa la forza relativa al loro voto di circa 1,34%.

VOTE WEIGHT	2.7000 EOS
DECAYED VOTE WEIGHT	1.7861 EOS

Decayed: 66.15%, next decay Saturday 00:00 UTC



Se invece si prova il voto, non c'è bisogno di rivotare per il proxy e se quest'ultimo non rivotano, il proprio account non subisce nessun effetto negativo, ma solo l'account del proxy sarà colpito dalla decadenza.

Purtroppo questo tipo di sistema funziona in parte perché non tutti

rivotano con quella frequenza o addirittura votano, dato che sono veramente pochissimi gli utenti che votano per i BP o per le proposte di referendum e quindi solo una piccola parte degli utenti ha il controllo dell'andamento dei BP quindi sotto quell'aspetto c'è ancora di strada da fare, ed è per questo motivo che le mie proposte di referendum che ho fatto mirano proprio ad aumentare il numero di votanti tramite un sistema di ricompensa per chi vota, così sarà incentivato a votare e votare spesso perché la mia proposta prende anche il sistema della decadenza del voto e quindi cercare di spingere l'utente ad essere più attivo sotto quel punto di

vista.

Come si diventa proxy con Cleos

system regproxy

Con questo comando possiamo registrare un account per diventare proxy, cioè votare per coloro che hanno delegato il loro voto al nostro account.

Positionals:

proxy *TEXT* - The proxy account to register

Options:

-h,--help Print this help message and exit

-x,--expiration *TEXT* - set the time in seconds before a transaction expires, defaults to 30s

-f,--force-unique - force the transaction to be unique. this will consume extra bandwidth and remove any protections against accidentally issuing the same transaction multiple times

-s,--skip-sign Specify if unlocked wallet keys should be used to sign transaction

-d,--dont-broadcast - Don't broadcast transaction to the network

(just print to stdout)

-r,--ref-block *TEXT* set the reference block num or block id used for TAPOS (Transaction as Proof-of-Stake)

-p,--permission *TEXT* - An account and permission level to authorize, as in 'account@permission' (defaults to 'account@active')

--max-cpu-usage-ms *UINT* - set an upper limit on the milliseconds of cpu usage budget, for the execution of the transaction (defaults to 0 which means no limit)

--max-net-usage *UINT* - set an upper limit on the net usage budget, in bytes, for the transaction (defaults to 0

which means no limit)

Usage:

```
cleos system regproxy someaccountl
```

system unregproxy

Con questo comando si elimina il relativo proxy da un account.

Positionals:

proxy *TEXT* - The proxy account to unregister

Options:

-h,--help Print this help message

and exit

-x,--expiration *TEXT* - set the time in seconds before a transaction expires, defaults to 30s

-f,--force-unique - force the transaction to be unique. this will consume extra bandwidth and remove any protections against accidentally issuing the same transaction multiple times

-s,--skip-sign Specify if unlocked wallet keys should be used to sign transaction

-d,--dont-broadcast - Don't broadcast transaction to the network (just print to stdout)

-r,--ref-block *TEXT* set the

reference block num or block id used for TAPOS (Transaction as Proof-of-Stake)

`-p,--permission TEXT` - An account and permission level to authorize, as in 'account@permission' (defaults to 'account@active')

`--max-cpu-usage-ms UINT` - set an upper limit on the milliseconds of cpu usage budget, for the execution of the transaction (defaults to 0 which means no limit)

`--max-net-usage UINT` - set an upper limit on the net usage budget, in bytes, for the transaction (defaults to 0 which means no limit)

Usage:

```
cleos system unregproxy someaccount1
```

Come si vota per un BP con Cleos

system listproducers

Restituisce una lista dei BP in formato testo o jsno, con i seguenti parametri:

- Producer account name

- Producer key

- Producer URL

- Producer's scaled votes

Positionals:

none

Options:

-h,--help - Print this help message and exit

-j,--json - Output in JSON format

-l,--limit *UINT* - The maximum number of rows to return

-L,--lower *TEXT* - Lower bound value of key, defaults to first

Usage:

cleos system listproducers

system voteproducer prods

Con questo comando si può votare per uno o più BP

Positionals:

voter *TEXT* - The voting account
producers *TEXT* ... - The
account(s) to vote for. All options
from this position and following will
be treated as the producer list.

Options:

-h,--help Print this help message
and exit

-x,--expiration *TEXT* - set the

time in seconds before a transaction expires, defaults to 30s

-f,--force-unique - force the transaction to be unique. this will consume extra bandwidth and remove any protections against accidentally issuing the same transaction multiple times

-s,--skip-sign Specify if unlocked wallet keys should be used to sign transaction

-d,--dont-broadcast - Don't broadcast transaction to the network (just print to stdout)

-r,--ref-block *TEXT* set the reference block num or block id used for TAPOS (Transaction as Proof-of-

Stake)

`-p,--permission TEXT` - An account and permission level to authorize, as in 'account@permission' (defaults to 'account@active')

`--max-cpu-usage-ms UINT` - set an upper limit on the milliseconds of cpu usage budget, for the execution of the transaction (defaults to 0 which means no limit)

`--max-net-usage UINT` - set an upper limit on the net usage budget, in bytes, for the transaction (defaults to 0 which means no limit)

Usage:

cleos system voteproducer prods
someproducer1 someproducer2
someproducer3 someproducer4

system voteproducer approve

Aggiunge uno BP alla lista dei BP votati.

Positionals:

voter *TEXT* - The voting account
producer *TEXT* - The account to
vote for

Options:

-h,--help Print this help message and exit

-x,--expiration *TEXT* - set the time in seconds before a transaction expires, defaults to 30s

-f,--force-unique - force the transaction to be unique. this will consume extra bandwidth and remove any protections against accidentally issuing the same transaction multiple times

-s,--skip-sign Specify if unlocked wallet keys should be used to sign transaction

-d,--dont-broadcast - Don't broadcast transaction to the network

(just print to stdout)

`-r,--ref-block TEXT` set the reference block num or block id used for TAPOS (Transaction as Proof-of-Stake)

`-p,--permission TEXT` - An account and permission level to authorize, as in 'account@permission' (defaults to 'account@active')

`--max-cpu-usage-ms UINT` - set an upper limit on the milliseconds of cpu usage budget, for the execution of the transaction (defaults to 0 which means no limit)

`--max-net-usage UINT` - set an upper limit on the net usage budget, in bytes, for the transaction (defaults to 0

which means no limit)

Usage:

cleos system voteproducer approve
someproducer 1

system voteproducer unapprove

Rimuove un BP dalla lista dei BP votati.

Positionals:

voter *TEXT* - The voting account
producer *TEXT* - The account to
remove from voted producers

Options:

-h,--help Print this help message and exit

-x,--expiration *TEXT* - set the time in seconds before a transaction expires, defaults to 30s

-f,--force-unique - force the transaction to be unique. this will consume extra bandwidth and remove any protections against accidentally issuing the same transaction multiple times

-s,--skip-sign Specify if unlocked wallet keys should be used to sign transaction

-d,--dont-broadcast - Don't broadcast transaction to the network

(just print to stdout)

`-r,--ref-block TEXT` set the reference block num or block id used for TAPOS (Transaction as Proof-of-Stake)

`-p,--permission TEXT` - An account and permission level to authorize, as in 'account@permission' (defaults to 'account@active')

`--max-cpu-usage-ms UINT` - set an upper limit on the milliseconds of cpu usage budget, for the execution of the transaction (defaults to 0 which means no limit)

`--max-net-usage UINT` - set an upper limit on the net usage budget, in bytes, for the transaction (defaults to 0

which means no limit)

Usage:

```
cleos system voteproducer unapprove  
someproducer1
```

system voteproducers proxy

Con questo comando deleghiamo il nostro voto ad un proxy.

Positionals:

voter *TEXT* - The voting account

proxy *TEXT* - The proxy account

Options:

-h,--help Print this help message and exit

-x,--expiration *TEXT* - set the time in seconds before a transaction expires, defaults to 30s

-f,--force-unique - Force the transaction to be unique. this will consume extra bandwidth and remove any protections against accidentally issuing the same transaction multiple times

-s,--skip-sign Specify if unlocked wallet keys should be used to sign transaction

-d,--dont-broadcast - Don't broadcast transaction to the network

(just print to stdout)

`-r,--ref-block TEXT` - Set the reference block num or block id used for TAPOS (Transaction as Proof-of-Stake)

`-p,--permission TEXT` - An account and permission level to authorize, as in 'account@permission' (defaults to 'account@active')

`--max-cpu-usage-ms UINT` - Set an upper limit on the milliseconds of cpu usage budget, for the execution of the transaction (defaults to 0 which means no limit)

`--max-net-usage UINT` - Set an upper limit on the net usage budget, in bytes, for the transaction (defaults to 0

which means no limit)

Usage:

```
cleos system voteproducer proxy  
accountname1 proxyaccount
```

Funzione whitelist e blacklist dei BP

Altra funzione che ricoprono i BP è

quella di creare whitelist e blacklist o di specifici smart contract oppure che riguardano uno specifico account. Basta aggiungere al file config.ini o tramite linea di comando, ovviamente nominando il tipo di contratto del caso, per esempio eosio.token, e verrà settato in automatico la black o la whitelist.

Se si setta una whitelist, questa ignorerà qualsiasi blacklist, quindi è esclusiva e dove si possono solo mandare transazioni a quel contratto, e se si aggiungono diverse whitelist allora solo a quelle transazioni si potranno mandare le transazioni. Poi se si vuole mettere un contratto nella blacklist allora anche gli altri BP devono settare la stessa blacklist e devono bloccare lo

stesso contratto sulle loro macchine e poi riavviare il nodo, perché solo in questo modo quella transazione non raggiungerà i blocchi, perché basta che solo un BP non abbia quel tipo di blocco e quindi se viene bloccato da un BP la transazione allora passa a quello successivo che potrebbe non avere quel blocco e quindi inserirla.

Poi abbiamo anche l'author whitelist e author blacklist che è simile alla precedente ma invece di controllare la destinazione del contratto, controlla l'origine dell'account cioè chi sta usando il permesso, dato che quando si manda una transazione questa registra chi ha effettuato la transazione e quindi chi l'ha firmata e di conseguenza avendo questa

informazione puoi prevenire che queste persone possono eseguire le transazioni ed anche in questo caso la whitelist ha la precedenza sulla blacklist ed anche in questo scenario ogni BP deve settare i parametri della blacklist.

REX (Resource Exchange)

Rex, abbreviazione di Resource Exchange, è un marketplace creato all'interno della blockchain di EOS che permette, senza nessun rischio, di cedere

e prendere in prestito risorse, CPU e NET, che vengono messe a disposizione dagli utenti stessi.

Il concetto si introdusse per la prima volta da un esponente di block.one, Daniel Larimer, nell'agosto del 2018^[10] dove spiegava le varie modalità e tutto il sistema.

L'esigenza di creare un sistema simile è stata dovuta al fatto che per creare una dApp sull'ecosistema di EOS è necessario disporre di molte risorse, CPU, RAM e NET e che queste a causa sia del prezzo che della scarsità per il fatto che vengono bloccate per far funzionare il tutto, erano inaccessibili per i piccoli sviluppatori.

REX introduce infatti un sistema di

prestati dove chi detiene i token EOS li cede alla piattaforma REX, dove questi vengono convertiti in REX token, e questi per tutto il tempo che rimarranno nella piattaforma genereranno un interesse in base sia all'ammontare dei token EOS messi a disposizione e sia per il tempo che questi sono stati bloccati, con l'aspetto positivo che non si perde la possibilità di votare e di partecipare ai vari snapshot e di conseguenza ai vari airdrop che avverranno in futuro.

Come funziona REX

Abbiamo detto che REX è una

piattaforma per cedere o prendere in prestito risorse per avere un ritorno per chi cede le stesse, mentre si paga un tot di EOS per prendere in prestito le risorse, dove REX permette contratti di prestito a 30 giorni

e il ritorno per chi cede queste risorse è determinato dalla domanda del network delle risorse, che può essere integrato dalle fee dell'asta dei nomi e dal trading della RAM, quindi sotto quel punto di vista non ci sono problemi per quanto riguarda la copertura degli interessi da distribuire ai partecipanti.

La piattaforma utilizza un token dedicato, il REX token, non trasferibile

e dove non si può fare trade dello stesso, e deve essere intesa più come una unità interna di misura della piattaforma REX.

Gli utenti possono acquistare REX cedendo le loro risorse o tramite i fondi liquidi che hanno nel loro account oppure trasferendo le risorse messe in stake, CPU e NET, direttamente senza prima fare l'unstake delle stesse.

Per evitare che ci sia una manipolazione del mercato, una volta che si è ricevuti questi token devono maturare dopo alcuni giorni e solo dopo che sono maturati allora si possono vendere; gli stessi maturano dopo 4 giorni e precisamente alle 00:00:00 UTC, che si sono acquistati.

Dopo che è passato questo periodo di maturazione si possono vendere immediatamente a meno che non c'è una scarsità della liquidità nel senso che non ci sono abbastanza EOS nella pool per coprire la vendita. Altra nota interessante del valore di REX, rispetto ad EOS, è che questo può solo aumentare ed è impossibile ricever di meno degli EOS che si sono investiti, ed inoltre quello che si accumula nel relativo account aumenterà e farà parte delle risorse che avremmo messo a disposizione.

Altra funzione che è stata integrata a protezione degli utenti è quella che riguarda la possibilità di mettere in un deposito questi token, che equivale a

metterli in stake sulla piattaforma REX, dove per prelevarli bisogna attendere un periodo di ulteriori 4 giorni, così nel caso le chiavi dell'account fossero state compromesse si avrebbe del tempo per sovrascrivere le vecchie chiavi dell'account.

Requisiti per utilizzare REX

Per utilizzare REX c'è solo un requisito da rispettare ed è quello di aver votato per almeno 21 BP oppure delegato il loro voto tramite un proxy, ed in questo modo si potrà operare con la piattaforma.

Prezzo di REX

Il prezzo dei relativi token di REX e del tasso con cui vengono calcolati per generare il prestito si basa su un'equazione Bancor^[11] e quindi non viene determinato dagli utenti che non potranno interferire con l'andamento del mercato e quindi devono adeguarsi.

A seguire una breve spiegazione nel dettaglio di come funzionano le equazioni utilizzate nel sistema REX, prima di tutto dobbiamo fare riferimento ai saldi della REX pool che sono rappresentati sia dai loro nomi delle variabili dello smart contract in C++ e dalle variabili matematiche usate per rappresentare le equazioni, inoltre dove

non è esplicito, tutte le unità del bilancio, i pagamenti delle fee e le risorse messe in stake sono con il token SYS del sistema.

I saldi sono:

- total_unlent, rappresenta il saldo SYS che è disponibile per prendere in affitto i token, ed utilizzeremo la lettera u per rappresentarla, $u = \text{total_unlent}$;

- total_lent, rappresenta il saldo SYS del totale dei token presi in affitto, ed è la somma dei token messi in stake dei prestiti aperti al momento, ed utilizzeremo la l per rappresentarla, $l = \text{total_lent}$;

- $total_rent$, è un saldo virtuale, dove il valore iniziale di questo saldo deve essere positivo, dove $total_rent$ e $total_unlent$ sono 2 fattori dell'algoritmo Bancor che determina il prezzo delle risorse CPU e NET, ed utilizzeremo la f per rappresentarla, $f = total_rent$.

Come calcolare i prestiti di REX

Per prendere in affitto risorse CPU e NET, il totale messo in stake da queste risorse per 30 giorni è calcolato con una funzione che misura la fee del prestito, Δf , ed i saldi della REX pool, $u = total_unlent$ e $f = total_rent$, utilizzando l'equazione Bancor e per un determinato

prestito gli diamo un nome i (equazione 1):

$$\Delta u^{(i)} = \Delta f^{(i)} \frac{u}{f + \Delta f^{(i)}}.$$

Per esempio, in un certo momento nel tempo, $u = 5 \times 10^7$, e $f = 3 \times 10^4$, le fee per il prestito con $\Delta f^{(i)} = 1$, portano il risultato a $\Delta u^{(i)} = 1666.6111$ di valore di token SYS, poi per calcolare il relativo tasso del prestito è dato da $r \approx f/u$, ed in questo caso $r \approx \Delta f^{(i)}/\Delta u^{(i)}$ ed è circa 0,06%.

Dopo che il prestito è stato creato, il saldo della REX pool viene aggiornato come di seguito (equazione 2):

$$\begin{aligned}u &\rightarrow u - \Delta u^{(i)} + \Delta f^{(i)}, \\l &\rightarrow l + \Delta u^{(i)}, \\f &\rightarrow f + \Delta f^{(i)}.\end{aligned}$$

Da notare che f è un saldo virtuale e che quindi non c'è il rischio della double spending aggiungendo $\Delta f^{(i)}$ sia a u che a f .

Inizializzare la REX pool

Inizialmente la Rex pool è vuota ($u = l = 0$), dove quando gli utenti che cedono i loro token SYS per comprare REX, la u aumenta, mentre invece il saldo f , dato che è virtuale deve essere inizializzato con un valore f_0 , ovviamente questo valore non deve essere 0 altrimenti il

primo prestito utilizzerà l'intero saldo u indipendentemente dalle fee. Questa verifica si può fare semplicemente settando $f = 0$ nell'equazione 1 che risulterà $\Delta u = u$ per ogni $\Delta f > 0$ ed in questo modo $f_0 > 0$.

Adesso bisogna decidere un valore pratico per f_0 , ed il saldo della REX pool u è previsto che raggiunga i 10 milioni dal lancio (attualmente sono oltre 100 milioni di token disponibili), quindi stimeremo $u_0 = 2 \times 10^7$ come valore di riferimento.

La risoluzione del prestito

Quando il prestito i termina, il

corrispondente delle risorse prese in prestito, $\Delta u^{(i)}$, sono rilasciate, quindi passano da $total_lent$ a $total_unlent$, ed il saldo f è aggiornato sottraendo il risultato dall'equazione inversa (equazione 3):

$$\Delta f^{(i)} = \Delta u^{(i)} \frac{f'}{u' + \Delta u^{(i)}}$$

Dove $\Delta u^{(i)}$ è calcolato usando l'equazione 1, mentre u' e f' sono i valori del prestito esaurito. Dato che questi valori sono differenti da quelli della creazione del prestito, allora avremo $f' \neq f$ e $u' \neq u$, ed anche $\Delta f^{(i)} \neq \Delta f^{(i)}$, dove l'output finale sarà espresso in questo modo (equazione 4):

$$\begin{aligned}
 u' &\rightarrow u' + \Delta u^{(i)}, \\
 l' &\rightarrow l' - \Delta u^{(i)}, \\
 f' &\rightarrow f' - \Delta f^{(i)}.
 \end{aligned}$$

Guardando all'equazione 3 possiamo notare che il tempo della risoluzione, `total_unlent`, è uguale a 0 ($u' = 0$), quindi l'equazione $\Delta f^{(i)} = f'$, di conseguenza nell'equazione 4 $f' = 0$ dopo che il prestito si è concluso. Questo scenario può essere dato se mentre c'è un prestito eccezionale, uno o più possessori di token REX potrebbero vendere abbastanza token REX da causare `total_unlent` un drop a 0, $u' = 0$ e per evitare che si raggiunga una cosa del genere si è imposto un limite minimo dinamico (dynamic lower bound).

Limite minimo saldo ulent

Per comodità chiamiamo ulb il limite minimo dinamico e che questo sarà quello relativo a u , quindi in un punto del tempo u sarà maggiore o uguale a ulb , $u \geq ulb$, poi dobbiamo definire che ulb è maggiore di 0 se ci sono contratti, $ulb > 0$ e poi se i prestiti sono terminati ulb sarà uguale a 0, $ulb = 0$. La seconda condizione permette ai possessori dei token REX di vendere i loro REX token, quindi settiamo uld ad una frazione di 1, per esempio $ulb = \alpha \times 1$, dove $0 < \alpha < 1$, soddisfa entrambi i requisiti, ed inoltre possiamo abbassare ulb così da non

causare degli ordini di vendita che si accumuleranno e che le azioni per prendere in prestito il tutto possa fallire, quindi settiamo $\alpha = 0.2$ ed in questo caso avremo $ulb = 0.2 \times l$.

Da notare che si è scelto di calcolare ulb in funzione di l invece che di f e per 2 ragioni, la prima perché u si aspetta che sia differente da f per ordini di grandezze che renderebbe la comparazione impraticabile e la seconda perché il valore di f non può essere usato per determinare se ci sono dei contratti attivi.

Aggiustare il saldo della REX pool virtuale

In caso in cui le condizioni iniziali non

hanno saldi è previsto di utilizzare un backup per risolvere il problema, dove questo può succedere, ad esempio, se dopo un periodo di tempo il `total_unlent` rimane al di sotto del valore di riferimento di $u_0 = 2 \times 10^7$ descritto in precedenza, e quindi significa che il tasso iniziale per i prestiti devono essere superiori ad un target di $r_0 \approx 0.1\%$ o determinato similamente dal tasso delle risorse del mercato.

Quest'azione permette ai producer di settare il saldo di `f` ad un determinato valore utilizzando l'equazione 2 con $f_0 \approx r_0 \times u$, dove `u` è il valore corrente di `total_unlent` e `r` è il target del tasso di costo per prendere in prestito i token.

Derivate delle equazioni

Il protocollo Bancor permette una liquidità istantanea collegando una riserva di valuta ad uno smart contract ed è definito la riserva frazionaria in questo modo:

$$F = \frac{R}{SP},$$

Dove R è il valore corrente della riserva di valore, S è la supply corrente del token e P è il prezzo corrente del token rapportato alla riserva di valore, quindi il protocollo determina F in modo sempre costante e detta il valore predeterminato del prezzo rispetto al

comportamento della supply.

Uno dei risultati del protocollo in un'equazione che determina l'ammontare da pagare per un dato numero di token dati è (equazione 5):

$$\Delta R = R_0 \left[\left(1 + \frac{\Delta S}{S_0} \right)^{\frac{1}{F}} - 1 \right],$$

Dove R_0 è il valore iniziale della riserva, S_0 è la supply iniziale del token e ΔS è il numero dei token richiesti.

Poi possiamo derivare anche l'equazione inversa (equazione 6):

$$\Delta S = S_0 \left[\left(1 + \frac{\Delta R}{R_0} \right)^F - 1 \right],$$

Questa determina il numero dei token richiesti per un determinato pagamento di una somma, e dopo che i token sono stati richiesti la supply viene aggiornata a $S = S_0 + \Delta S$ e la riserva a $R = R_0 + \Delta R$.

Ora consideriamo che il token sia connesso a 2 riserve $R^{(1)}$ e $R^{(2)}$ e assumiamo che il tasso della riserva frazionaria del token sia la stessa per entrambe le riserve, quindi al pagamento di $\Delta R^{(1)}$, risulterà una richiesta di token utilizzando l'equazione 6 applicata a $R^{(1)}$ (equazione 7):

$$\Delta S = S_0 \left[\left(1 + \frac{\Delta R^{(1)}}{R_0^{(1)}} \right)^F - 1 \right] \implies \frac{S_0 + \Delta S}{S_0} = \left(\frac{R_0^{(1)} + \Delta R^{(1)}}{R_0^{(1)}} \right)^F.$$

In questa, i token poi sono venduti (dove bisogna aggiungere $-\Delta S$ alla supply del token), in cambio della seconda riserva di valuta ottenendo (equazione 8):

$$\Delta R^{(2)} = R_0^{(2)} \left[\left(1 - \frac{\Delta S}{S} \right)^{\frac{1}{p}} - 1 \right] = R_0^{(2)} \left[\left(\frac{S_0}{S_0 + \Delta S} \right)^{\frac{1}{p}} - 1 \right].$$

Dove infine rimpiazzeremo l'equazione 7 nell'equazione 8:

$$\Delta R^{(2)} = -R_0^{(2)} \frac{\Delta R^{(1)}}{R_0^{(1)} + \Delta R^{(1)}}.$$

Da notare che $\Delta R^{(2)}$ e $\Delta R^{(1)}$ hanno segno opposto, e che le 2 riserve sono $f \equiv R^{(1)}$ e $u \equiv R^{(2)}$.

Utilizzare REX con un'interfaccia web

Per utilizzare REX ci sono diversi metodi e piattaforme od anche wallet EOS avanzati che permettono, collegandosi con il proprio account EOS, di gestire tutta la procedura con una semplice interfaccia grafica, nei passaggi seguenti utilizzeremo il block explorer blocks.io che permette questa procedura.

Prima di tutto bisogna recarsi alla pagina web in questione, loggarsi con il proprio account a seconda del metodo che si preferisce, per esempio con Scatter, e cercare la voce o menù REX:

The screenshot shows the top navigation bar of Bloks.io with the logo on the left and navigation links for Monitor, Wallet, Vote, dApps, Tokens, DEX, and REX. The REX link is circled in red. Below the navigation is a search bar and a 'Connect To Wallet' section with a grid of wallet providers: Scatter, EOS Auth, Lynx, Trezor, Ledger S/X, SimpleEOS, EOStock, cleos, Math/Leaf, Keycat, Anchor, and eosc. At the bottom of the grid, there is a link to 'Create A New Wallet' and a note about accepting terms.

Nella pagina successiva avremo le statistiche di REX e le informazioni del nostro wallet, dove possiamo osservare la liquidità di REX, il tasso di interesse a 30 giorni o annuale, il totale di EOS che possiamo prendere in prestito con 1 EOS, il totale di EOS presi in prestito ed il prezzo di REX in EOS:

Wallet

Liquid Balance:
Please log in

Liquid Loans:
Please log in

Processing Loans:
Please log in

Savings Loans:
Please log in

Total in REX:
Please log in

Lifetime ROI:
Please log in

REX Stats

REX Liquidity:
103,997,615.1866 EOS

30 days/1 year interest:
0.0157%/0.1892%

Resource Price:
1 EOS Can Borrow 4,752.2543 EOS for 30 days

Total Borrowed:
2,010,758.6603 EOS (1.93%)

REX Price:
0.00010017 EOS / 1 REX

Poi abbiamo diverse tab che permettono di operare con REX, ed iniziamo con la pagina Lend Liquid EOS che permette di cedere i nostri EOS liquidi (non in stake) che abbiamo sull'account e decidere quanto cedere, oppure richiedere gli EOS ceduti (unlend EOS), poi una volta inserito il tutto confermiamo la transazione, ricordiamo che questa procedura è necessario aspettare 4 giorni prima che il tutto parta:

Lend EOS Can Lend: 0.0004 EOS

Amount: 0.0004 EOS
25% 50% 75% 100%

[Log In to lend EOS](#)

Estimated Interest Rates

30 Day:	0.0157%
APR:	0.1892%

Unlend EOS

Can Unlend: 0.0000 EOS

Amount: 0.0000 EOS
25% 50% 75% 100%

[Log In to unlend EOS](#)

Loans Processing

Amount Can Unlend on:

No loans currently being processed.

La successiva pagina invece opera tramite gli EOS che abbiamo in stake ed in questo caso il tutto sarà immediatamente operativo e non ci sarà bisogno di fare l'unstake dei token, come la solito dobbiamo sempre confermare la transazione:

Lend Staked EOS

Lend Staked CPU and NET to REX directly, without waiting 3 days for unstaking.

EOS Staked

To	Lend Staked CPU (EOS)	Lend Staked NET (EOS)
cryptonauta1	2.2 / 2.2	0.5 / 0.5

Lend 2.000 Staked EOS to REX

Successivamente abbiamo la pagina dove noi possiamo richiedere CPU e

NET, dove inseriamo i vari parametri dell'ammontare che necessitiamo e poi confermiamo il tutto e poi ci sarà un breve ricapitolo delle informazioni di quanto preso in prestito:

Borrow CPU and NET

Can Borrow: 0.0004 EOS

Borrow For:	<input type="text" value="e.g. eosio"/>	Borrow for yourself
CPU Amount:	<input type="text" value="0"/> EOS	<input type="radio"/> 25% <input type="radio"/> 50% <input type="radio"/> 75% <input type="radio"/> 100%
NET Amount:	<input type="text" value="0"/> EOS	<input type="radio"/> 25% <input type="radio"/> 50% <input type="radio"/> 75% <input type="radio"/> 100%
Auto Renew Fund:	<input type="text" value="0"/> EOS	<input type="radio"/> 25% <input type="radio"/> 50% <input type="radio"/> 75% <input type="radio"/> 100%
CPU Borrowed:	0 EOS	
Net Borrowed:	0 EOS	
Total:	0 EOS (0.0000 Loan Fee + 0.0000 Renew Fund)	
Current Rate:	1 EOS Can Borrow 4,752.3031 EOS for 30 days	

[Log In to borrow EOS](#)

CPU Loans

From	To	Borrowed	Loan Fee	Auto-Renew Fund	Expiration
------	----	----------	----------	-----------------	------------

NET Loans

From	To	Borrowed	Loan Fee	Auto-Renew Fund	Expiration
------	----	----------	----------	-----------------	------------

Poi abbiamo un piccolo report con le statistiche degli ultimi 7 giorni riguardo l'andamento di REX:



Total # of Loans:	18362
Total REX:	1,038,220,617,952,7050 REX
Total Lendable:	103,998,041,6652 EOS
Total Borrowed:	2,010,447,6436 EOS
Total Unborrowed:	101,987,594,0216 EOS
Borrowed Percent:	1.93%
Total Interest:	1,459,6667 EOS
REX Price:	0.00010017 REX / 1 EOS
Resource Price:	1 EOS can borrow 4,752.3031 EOS for 30 days

Altra pagina importante è quella che riguarda la possibilità di mettere in stake, saving, i token REX e quindi utilizziamo questa pagina per decidere se e quanto mettere nel fondo saving oppure prelevare dal suddetto fondo, e poi confermiamo la transazione:

Move to or from savings:

Moving processing/liquid loans to savings adds a 4 day timer to selling your loans. It does not give you extra rewards.

Move lent EOS to savings

Can move: 0.0000 EOS

Amount: EOS

25%

50%

75%

100%

[Log in to move EOS to Savings](#)**Move lent EOS from savings**

Can remove: 0.0000 EOS

Amount: EOS

25%

50%

75%

100%

[Log in to move EOS to savings](#)

Ed infine c'è la pagina relativa ad uno storico di tutte le nostre transazioni relative a REX così da poterle vedere in modo chiaro:

Please log in to view REX history.

Utilizzare REX con cleos

Ovviamente è possibile operare anche con REX tramite l'interfaccia cleos con i relativi comandi.

system rex deposit

Deposita i token liquidi dall'account del proprietario al proprio saldo rex.

Positionals:

owner TEXT - l'account dove ci sono i fondi REX (richiesto)

amount TEXT - l'ammontare da depositare nei fondi REX (richiesto)

Options:

- h, --help - stampa un messaggio di aiuto ed esce
- x, --expiration TEXT - setta il tempo in secondi prima che il contratto finisca, di default è 30 secondi
- f, --force-unique - forza la transazione ad essere unica, questo consumerà banda extra e rimuoverà qualsiasi protezione contro errori multipli provenienti dalla stessa transazione
- s, --skip-sign - specifica se un wallet sbloccato può essere usato per firmare la transazione
- d, --dont-broadcast - non trasmette la transazione alla rete (la stampa al stdout)

-r, --ref-block TEXT - setta i riferimenti del numero del blocco o l'id del blocco usato per la TAPOS

-p, --permission TEXT - un account ed un livello di permesso, come in account@permission (default è account@active)

--max-cpu-usage-ms UINT - setta un tetto massimo di millisecondi per l'utilizzo della cpu, per l'esecuzione del contratto (default è 0 cioè senza limiti)

--max-net-usage UINT - setta un tetto massimo di net da usare, in bytes, per la transazione (default è 0 cioè non ci sono limiti)

Usage:

esempio

```
cleos system rex deposit accountname1  
"1 SYS"
```

system rex withdraw

Preleva dal fondo REX del proprietario e lo trasferisce nel saldo liquido dell'account del proprietario.

Positionals:

owner TEXT - l'account dove ci sono i fondi REX (richiesto)

amount TEXT - l'ammontare da depositare nei fondi REX (richiesto)

Options:

-h, --help - stampa un messaggio di aiuto ed esce

-x, --expiration TEXT - setta il tempo in secondi prima che il contratto finisca, di default è 30 secondi

-f, --force-unique - forza la transazione ad essere unica, questo consumerà banda extra e rimuoverà qualsiasi protezione contro errori multipli provenienti dalla stessa transazione

-s, --skip-sign - specifica se un wallet sbloccato può essere usato per firmare la transazione

-d, --dont-broadcast - non trasmette la transazione alla rete (la stampa al

stdout)

-r, --ref-block TEXT - setta i riferimenti del numero del blocco o l'id del blocco usato per la TAPOS

-p, --permission TEXT - un account ed un livello di permesso, come in account@permission (default è account@active)

--max-cpu-usage-ms UINT - setta un tetto massimo di millisecondi per l'utilizzo della cpu, per l'esecuzione del contratto (default è 0 cioè senza limiti)

--max-net-usage UINT - setta un tetto massimo di net da usare, in bytes, per la transazione (default è 0 cioè non ci sono limiti)

Usage:

esempio

```
cleos system rex withdraw  
accountname1 "1 SYS"
```

system rex buyrex

Permette di comprare token REX tramite i fondi EOS in REX.

Positionals:

from TEXT - l'account che deve comprare i token REX (richiesto)
amount TEXT - l'ammontare da prendere dal fondo REX per comprare token REX

(richiesto)

Options:

-h, --help - stampa un messaggio di aiuto ed esce

-x, --expiration TEXT - setta il tempo in secondi prima che il contratto finisca, di default è 30 secondi

-f, --force-unique - forza la transazione ad essere unica, questo consumerà banda extra e rimuoverà qualsiasi protezione contro errori multipli provenienti dalla stessa transazione

-s, --skip-sign - specifica se un wallet sbloccato può essere usato per firmare la transazione

-d, --dont-broadcast - non trasmette la transazione alla rete (la stampa al stdout)

-r, --ref-block TEXT - setta i riferimenti del numero del blocco o l'id del blocco usato per la TAPOS

-p, --permission TEXT - un account ed un livello di permesso, come in account@permission (default è account@active)

--max-cpu-usage-ms UINT - setta un tetto massimo di millisecondi per l'utilizzo della cpu, per l'esecuzione del contratto (default è 0 cioè senza limiti)

--max-net-usage UINT - setta un tetto massimo di net da usare, in bytes, per la transazione (default è 0 cioè non ci sono limiti)

Usage:

esempio

```
cleos system rex buyrex accountname1  
"1 REX"
```

system rex lendrex

Depositare i token al fondo REX ed usare i token per comprare REX.

Positionals:

from TEXT - l'account che deve comprare i token REX (richiesto)

amount TEXT - l'ammontare da prendere dal fondo REX per comprare token REX (richiesto)

Options:

-h, --help - stampa un messaggio di aiuto ed esce

-x, --expiration TEXT - setta il tempo in secondi prima che il contratto finisca, di default è 30 secondi

-f, --force-unique - forza la transazione ad essere unica, questo consumerà banda extra e rimuoverà qualsiasi protezione contro errori multipli provenienti dalla stessa transazione

-s, --skip-sign - specifica se un wallet sbloccato può essere usato per firmare la transazione

-d, --dont-broadcast - non trasmette la transazione alla rete (la stampa al stdout)

-r, --ref-block TEXT - setta i riferimenti del numero del blocco o l'id del blocco usato per la TAPOS

-p, --permission TEXT - un account ed un livello di permesso, come in account@permission (default è account@active)

--max-cpu-usage-ms UINT - setta un tetto massimo di millisecondi per l'utilizzo della cpu, per l'esecuzione del contratto (default è 0 cioè senza limiti)

--max-net-usage UINT - setta un tetto

massimo di net da usare, in bytes, per la transazione (default è 0 cioè non ci sono limiti)

Usage:

esempio

```
cleos system rex lendrex accountname1  
"1 REX"
```

system rex cancelrexorder

Se non c'è un ordine ancora completato i sellrex allora possiamo cancellare quell'ordine.

Positionals:

owner TEXT - l'account proprietario per vendere l'ordine (richiesto)

Options:

-h, --help - stampa un messaggio di aiuto ed esce

-x, --expiration TEXT - setta il tempo in secondi prima che il contratto finisca, di default è 30 secondi

-f, --force-unique - forza la transazione ad essere unica, questo consumerà banda extra e rimuoverà qualsiasi protezione contro errori multipli provenienti dalla stessa transazione

-s, --skip-sign - specifica se un wallet sbloccato può essere usato per firmare la transazione

-d, --dont-broadcast - non trasmette la transazione alla rete (la stampa al stdout)

-r, --ref-block TEXT - setta i riferimenti del numero del blocco o l'id del blocco usato per la TAPOS

-p, --permission TEXT - un account ed un livello di permesso, come in account@permission (default è account@active)

--max-cpu-usage-ms UINT - setta un tetto massimo di millisecondi per l'utilizzo della cpu, per l'esecuzione del contratto (default è 0 cioè senza limiti)

--max-net-usage UINT - setta un tetto

massimo di net da usare, in bytes, per la transazione (default è 0 cioè non ci sono limiti)

Usage:

esempio

```
cleos      system      rex      cancelorder  
accountname l
```

system rex mvto savings

Sposta I token REX a fondo savings

owner TEXT - l'account dove ci sono i

fondi REX (richiesto)

rex TEXT - il totale dei token REX che saranno spostati al fondo savings

Options:

-h, --help - stampa un messaggio di aiuto ed esce

-x, --expiration TEXT - setta il tempo in secondi prima che il contratto finisca, di default è 30 secondi

-f, --force-unique - forza la transazione ad essere unica, questo consumerà banda extra e rimuoverà qualsiasi protezione contro errori multipli provenienti dalla stessa transazione

-s, --skip-sign - specifica se un wallet sbloccato può essere usato per firmare

la transazione

-d, --dont-broadcast - non trasmette la transazione alla rete (la stampa al stdout)

-r, --ref-block TEXT - setta i riferimenti del numero del blocco o l'id del blocco usato per la TAPOS

-p, --permission TEXT - un account ed un livello di permesso, come in account@permission (default è account@active)

--max-cpu-usage-ms UINT - setta un tetto massimo di millisecondi per l'utilizzo della cpu, per l'esecuzione del contratto (default è 0 cioè senza limiti)

--max-net-usage UINT - setta un tetto massimo di net da usare, in bytes, per la transazione (default è 0 cioè non ci sono

limiti)

Usage:

esempio

```
cleos      system      rex      mvto savings  
accountname1 "1 REX"  
system rex mvfrom savings
```

Sposta I token REX fuori dal fondo savings

owner TEXT - l'account dove ci sono i fondi REX (richiesto)

rex TEXT - il totale dei token REX che saranno spostati fuori dal fondo savings

Options:

- h, --help - stampa un messaggio di aiuto ed esce
- x, --expiration TEXT - setta il tempo in secondi prima che il contratto finisca, di default è 30 secondi
- f, --force-unique - forza la transazione ad essere unica, questo consumerà banda extra e rimuoverà qualsiasi protezione contro errori multipli provenienti dalla stessa transazione
- s, --skip-sign - specifica se un wallet sbloccato può essere usato per firmare la transazione
- d, --dont-broadcast - non trasmette la transazione alla rete (la stampa al stdout)

-r, --ref-block TEXT - setta i riferimenti del numero del blocco o l'id del blocco usato per la TAPOS

-p, --permission TEXT - un account ed un livello di permesso, come in account@permission (default è account@active)

--max-cpu-usage-ms UINT - setta un tetto massimo di millisecondi per l'utilizzo della cpu, per l'esecuzione del contratto (default è 0 cioè senza limiti)

--max-net-usage UINT - setta un tetto massimo di net da usare, in bytes, per la transazione (default è 0 cioè non ci sono limiti)

Usage:

esempio

```
cleos system rex mvfromsavings  
accountname1 "1 REX"  
system rex rentcpu
```

Prende in prestito la banda CPU per 30 giorni.

Positionals:

from TEXT - l'account che deve comprare I token REX (richiesto)
receiver TEXT - l'account che riceverà la banda delle CPU (richiesto)
loan_payment TEXT - le fee del prestito che dovranno essere pagate, usato per calcolare il totale della banda presa in

prestito (richiesto)

loan_fund TEXT - il fondo del prestito che sarà usato per rinnovare in maniera automatico il tutto, può essere 0 (richiesto)

Options:

-h, --help - stampa un messaggio di aiuto ed esce

-x, --expiration TEXT - setta il tempo in secondi prima che il contratto finisca, di default è 30 secondi

-f, --force-unique - forza la transazione ad essere unica, questo consumerà banda extra e rimuoverà qualsiasi protezione contro errori multipli provenienti dalla stessa transazione

-s, --skip-sign - specifica se un wallet sbloccato può essere usato per firmare la transazione

-d, --dont-broadcast - non trasmette la transazione alla rete (la stampa al stdout)

-r, --ref-block TEXT - setta i riferimenti del numero del blocco o l'id del blocco usato per la TAPOS

-p, --permission TEXT - un account ed un livello di permesso, come in account@permission (default è account@active)

--max-cpu-usage-ms UINT - setta un tetto massimo di millisecondi per l'utilizzo della cpu, per l'esecuzione del contratto (default è 0 cioè senza limiti)

--max-net-usage UINT - setta un tetto

massimo di net da usare, in bytes, per la transazione (default è 0 cioè non ci sono limiti)

Usage:

esempio

```
cleos system rex rentcpu accountname1  
accountname2 "1 EOS" 0
```

system rex rentnet

Prende in prestito la banda NET per 30 giorni.

Positionals:

from TEXT - l'account che deve comprare I token REX (richiesto)
receiver TEXT - l'account che riceverà la banda delle NET (richiesto)
loan_payment TEXT - le fee del prestito che dovranno essere pagate, usato per calcolare il totale della banda presa in prestito (richiesto)
loan_fund TEXT - il fondo del prestito che sarà usato per rinnovare in maniera automatico il tutto, può essere 0 (richiesto)

Options:

-h, --help - stampa un messaggio di aiuto ed esce

-x, --expiration TEXT - setta il tempo in secondi prima che il contratto finisca, di default è 30 secondi

-f, --force-unique - forza la transazione ad essere unica, questo consumerà banda extra e rimuoverà qualsiasi protezione contro errori multipli provenienti dalla stessa transazione

-s, --skip-sign - specifica se un wallet sbloccato può essere usato per firmare la transazione

-d, --dont-broadcast - non trasmette la transazione alla rete (la stampa al stdout)

-r, --ref-block TEXT - setta i riferimenti del numero del blocco o l'id del blocco

usato per la TAPOS

-p, --permission TEXT - un account ed un livello di permesso, come in account@permission (default è account@active)

--max-cpu-usage-ms UINT - setta un tetto massimo di millisecondi per l'utilizzo della cpu, per l'esecuzione del contratto (default è 0 cioè senza limiti)

--max-net-usage UINT - setta un tetto massimo di net da usare, in bytes, per la transazione (default è 0 cioè non ci sono limiti)

Usage:

esempio

```
cleos system rex rentnet accountname1  
accountname2 "1 EOS" 0
```

system rex fundcpuloan

Deposita dentro il fondo del prestito della CPU.

Positionals:

from TEXT - proprietario del prestito (richiesto)

loan_num TEXT - l'ID del prestito (richiesto)

payment TEXT - il totale da depositare (richiesto)

Options:

- h, --help - stampa un messaggio di aiuto ed esce
- x, --expiration TEXT - setta il tempo in secondi prima che il contratto finisca, di default è 30 secondi
- f, --force-unique - forza la transazione ad essere unica, questo consumerà banda extra e rimuoverà qualsiasi protezione contro errori multipli provenienti dalla stessa transazione
- s, --skip-sign - specifica se un wallet sbloccato può essere usato per firmare la transazione
- d, --dont-broadcast - non trasmette la transazione alla rete (la stampa al stdout)

-r, --ref-block TEXT - setta i riferimenti del numero del blocco o l'id del blocco usato per la TAPOS

-p, --permission TEXT - un account ed un livello di permesso, come in account@permission (default è account@active)

--max-cpu-usage-ms UINT - setta un tetto massimo di millisecondi per l'utilizzo della cpu, per l'esecuzione del contratto (default è 0 cioè senza limiti)

--max-net-usage UINT - setta un tetto massimo di net da usare, in bytes, per la transazione (default è 0 cioè non ci sono limiti)

Usage:

esempio

```
cleos      system      rex      fundcpuloan  
accountname1 abc123 "1 EOS"
```

system rex fundnetloan

Deposita dentro il fondo del prestito della NET.

Positionals:

from TEXT - proprietario del prestito (richiesto)

loan_num TEXT - l'ID del prestito

(richiesto)

payment TEXT - il totale da depositare

(richiesto)

Options:

-h, --help - stampa un messaggio di aiuto ed esce

-x, --expiration TEXT - setta il tempo in secondi prima che il contratto finisca, di default è 30 secondi

-f, --force-unique - forza la transazione ad essere unica, questo consumerà banda extra e rimuoverà qualsiasi protezione contro errori multipli provenienti dalla stessa transazione

-s, --skip-sign - specifica se un wallet sbloccato può essere usato per firmare

la transazione

-d, --dont-broadcast - non trasmette la transazione alla rete (la stampa al stdout)

-r, --ref-block TEXT - setta i riferimenti del numero del blocco o l'id del blocco usato per la TAPOS

-p, --permission TEXT - un account ed un livello di permesso, come in account@permission (default è account@active)

--max-cpu-usage-ms UINT - setta un tetto massimo di millisecondi per l'utilizzo della cpu, per l'esecuzione del contratto (default è 0 cioè senza limiti)

--max-net-usage UINT - setta un tetto massimo di net da usare, in bytes, per la transazione (default è 0 cioè non ci sono

limiti)

Usage:

esempio

```
cleos system rex fundnetloan  
accountname1 abc123 "1 EOS"
```

system rex defundcpuloan

Preleva dal fondo del prestito della CPU.

Positionals:

from TEXT - proprietario del prestito

(richiesto)

loan_num TEXT - l'ID del prestito

(richiesto)

payment TEXT - il totale da depositare

(richiesto)

Options:

-h, --help - stampa un messaggio di aiuto ed esce

-x, --expiration TEXT - setta il tempo in secondi prima che il contratto finisca, di default è 30 secondi

-f, --force-unique - forza la transazione ad essere unica, questo consumerà banda extra e rimuoverà qualsiasi protezione contro errori multipli provenienti dalla stessa transazione

-s, --skip-sign - specifica se un wallet sbloccato può essere usato per firmare la transazione

-d, --dont-broadcast - non trasmette la transazione alla rete (la stampa al stdout)

-r, --ref-block TEXT - setta i riferimenti del numero del blocco o l'id del blocco usato per la TAPOS

-p, --permission TEXT - un account ed un livello di permesso, come in account@permission (default è account@active)

--max-cpu-usage-ms UINT - setta un tetto massimo di millisecondi per l'utilizzo della cpu, per l'esecuzione del contratto (default è 0 cioè senza limiti)

--max-net-usage UINT - setta un tetto

massimo di net da usare, in bytes, per la transazione (default è 0 cioè non ci sono limiti)

Usage:

esempio

```
cleos system rex defundcpuloan  
accountname1 abc123 "1 EOS"
```

system rex defundnetloan

Preleva dal fondo del prestito della

NET.

Positionals:

from TEXT - proprietario del prestito
(richiesto)

loan_num TEXT - l'ID del prestito
(richiesto)

payment TEXT - il totale da depositare
(richiesto)

Options:

-h, --help - stampa un messaggio di aiuto
ed esce

-x, --expiration TEXT - setta il tempo in
secondi prima che il contratto finisca, di
default è 30 secondi

-f, --force-unique - forza la transazione ad essere unica, questo consumerà banda extra e rimuoverà qualsiasi protezione contro errori multipli provenienti dalla stessa transazione

-s, --skip-sign - specifica se un wallet sbloccato può essere usato per firmare la transazione

-d, --dont-broadcast - non trasmette la transazione alla rete (la stampa al stdout)

-r, --ref-block TEXT - setta i riferimenti del numero del blocco o l'id del blocco usato per la TAPOS

-p, --permission TEXT - un account ed un livello di permesso, come in account@permission (default è account@active)

--max-cpu-usage-ms UINT - setta un tetto massimo di millisecondi per l'utilizzo della cpu, per l'esecuzione del contratto (default è 0 cioè senza limiti)

--max-net-usage UINT - setta un tetto massimo di net da usare, in bytes, per la transazione (default è 0 cioè non ci sono limiti)

Usage:

esempio

```
cleos system rex defundnetloan  
accountname1 abc123 "1 EOS"
```

system rex consolidate

Dato che i token REX necessitano di 4 giorni prima che maturino, possiamo avere fino a 4 bucket allo stesso tempo, ognuno che consumerà il suo ammontare di RAM e se vogliamo liberare la RAM allora dovremmo consolidare il tutto in un unico bucket che matureranno da questo momento in poi e si usa l'azione consolidate.

Positionals:

owner TET - l'account proprietario che ha i fondi REX (richiesto)

Options:

-h, --help - stampa un messaggio di aiuto ed esce

-x, --expiration TEXT - setta il tempo in secondi prima che il contratto finisca, di default è 30 secondi

-f, --force-unique - forza la transazione ad essere unica, questo consumerà banda extra e rimuoverà qualsiasi protezione contro errori multipli provenienti dalla stessa transazione

-s, --skip-sign - specifica se un wallet sbloccato può essere usato per firmare la transazione

-d, --dont-broadcast - non trasmette la transazione alla rete (la stampa al stdout)

-r, --ref-block TEXT - setta i riferimenti del numero del blocco o l'id del blocco

usato per la TAPOS

-p, --permission TEXT - un account ed un livello di permesso, come in account@permission (default è account@active)

--max-cpu-usage-ms UINT - setta un tetto massimo di millisecondi per l'utilizzo della cpu, per l'esecuzione del contratto (default è 0 cioè senza limiti)

--max-net-usage UINT - setta un tetto massimo di net da usare, in bytes, per la transazione (default è 0 cioè non ci sono limiti)

Usage:

esempio

cleos system rex consolidate
accountname1

system rex updaterex

Aggiorna lo stake ed il peso dei voti del proprietario dei REX, risultato che può essere anche raggiunto rivotando normalmente.

Positionals:

owner TEXT - l'account proprietario che ha i fondi REX (richiesto)

Options:

-h, --help - stampa un messaggio di aiuto ed esce

-x, --expiration TEXT - setta il tempo in secondi prima che il contratto finisca, di default è 30 secondi

-f, --force-unique - forza la transazione ad essere unica, questo consumerà banda extra e rimuoverà qualsiasi protezione contro errori multipli provenienti dalla stessa transazione

-s, --skip-sign - specifica se un wallet sbloccato può essere usato per firmare la transazione

-d, --dont-broadcast - non trasmette la transazione alla rete (la stampa al stdout)

-r, --ref-block TEXT - setta i riferimenti

del numero del blocco o l'id del blocco
usato per la TAPOS

-p, --permission TEXT - un account ed
un livello di permesso, come in
account@permission (default è
account@active)

--max-cpu-usage-ms UINT - setta un
tetto massimo di millisecondi per
l'utilizzo della cpu, per l'esecuzione del
contratto (default è 0 cioè senza limiti)

--max-net-usage UINT - setta un tetto
massimo di net da usare, in bytes, per la
transazione (default è 0 cioè non ci sono
limiti)

Usage:

esempio

cleos system rex updatere
accountname1

system rex rexexec

Fa una manutenzione di REX processando i prestiti scaduti e gli ordini non ancora completati, aggiornando il tutto praticamente.

Positionals:

user TEXT - l'utente che esegue l'azione

(richiesto)

max TEXT - il numero massimo di prestiti CPU, di NET e gli ordini di vendita da processare

Options:

-h, --help - stampa un messaggio di aiuto ed esce

-x, --expiration TEXT - setta il tempo in secondi prima che il contratto finisca, di default è 30 secondi

-f, --force-unique - forza la transazione ad essere unica, questo consumerà banda extra e rimuoverà qualsiasi protezione contro errori multipli provenienti dalla stessa transazione

-s, --skip-sign - specifica se un wallet

sbloccato può essere usato per firmare la transazione

-d, --dont-broadcast - non trasmette la transazione alla rete (la stampa al stdout)

-r, --ref-block TEXT - setta i riferimenti del numero del blocco o l'id del blocco usato per la TAPOS

-p, --permission TEXT - un account ed un livello di permesso, come in account@permission (default è account@active)

--max-cpu-usage-ms UINT - setta un tetto massimo di millisecondi per l'utilizzo della cpu, per l'esecuzione del contratto (default è 0 cioè senza limiti)

--max-net-usage UINT - setta un tetto massimo di net da usare, in bytes, per la

transazione (default è 0 cioè non ci sono limiti)

Usage:

esempio

```
cleos system rex rexexec accountname1  
10
```

system rex closerex

Questa azione controlla prima se nell'account ci sono dei REX token, dei prestiti aperti per la CPU e la NET, o se ci sono dei token EOS nel fondo REX; se tutti questi parametri sono

vuoti, allora è possibile liberare la memoria RAM utilizzata per salvare i relativi dati nel contratto eosio.rex.

Positionals:

owner TExT - l'account proprietario che ha i fondi REX (richiesto)

Options:

-h, --help - stampa un messaggio di aiuto ed esce

-x, --expiration TEXT - setta il tempo in secondi prima che il contratto finisca, di default è 30 secondi

-f, --force-unique - forza la transazione ad essere unica, questo consumerà banda

extra e rimuoverà qualsiasi protezione contro errori multipli provenienti dalla stessa transazione

-s, --skip-sign - specifica se un wallet sbloccato può essere usato per firmare la transazione

-d, --dont-broadcast - non trasmette la transazione alla rete (la stampa al stdout)

-r, --ref-block TEXT - setta i riferimenti del numero del blocco o l'id del blocco usato per la TAPOS

-p, --permission TEXT - un account ed un livello di permesso, come in account@permission (default è account@active)

--max-cpu-usage-ms UINT - setta un tetto massimo di millisecondi per

l'utilizzo della cpu, per l'esecuzione del contratto (default è 0 cioè senza limiti)
--max-net-usage UINT - setta un tetto massimo di net da usare, in bytes, per la transazione (default è 0 cioè non ci sono limiti)

Usage:

esempio

```
cleos system rex closerex accountname1
```

system rex unstaketorex

Compra REX usando I token messi in stake.

from TEXT - l'account che deve comprare i token REX (richiesto)
receiver TEXT - l'account che deve ricevere i token in stake (richiesto)
from_cpu TEXT - il totale delle risorse CPU da togliere dallo stake per usarle per comprare i token REX (richiesto)
from_net TEXT - il totale delle risorse NET da togliere dallo stake per usarle per comprare i token REX (richiesto)

Options:

- h, --help - stampa un messaggio di aiuto ed esce
- x, --expiration TEXT - setta il tempo in secondi prima che il contratto finisca, di

default è 30 secondi

-f, --force-unique - forza la transazione ad essere unica, questo consumerà banda extra e rimuoverà qualsiasi protezione contro errori multipli provenienti dalla stessa transazione

-s, --skip-sign - specifica se un wallet sbloccato può essere usato per firmare la transazione

-d, --dont-broadcast - non trasmette la transazione alla rete (la stampa al stdout)

-r, --ref-block TEXT - setta i riferimenti del numero del blocco o l'id del blocco usato per la TAPOS

-p, --permission TEXT - un account ed un livello di permesso, come in account@permission (default è

account@active)

--max-cpu-usage-ms UINT - setta un tetto massimo di millisecondi per l'utilizzo della cpu, per l'esecuzione del contratto (default è 0 cioè senza limiti)

--max-net-usage UINT - setta un tetto massimo di net da usare, in bytes, per la transazione (default è 0 cioè non ci sono limiti)

Usage:

esempio

```
cleos system rex unstaketorex  
accountname1 accountname2 "1 EOS" 0
```

system rex sellrex

Vende I token REX per token EOS e se non c'è liquidità immediata allora l'ordine di vendita viene messo in coda e se precedentemente era aperto un ordine di vendita, richiamando quest'azione di nuovo combinerà i 2 ordini di vendita.

from TEXT - l'account che deve vendere I token REX (richiesto)

rex TEXT - il totale di token REX che devono essere venduti (richiesto)

Options:

-h, --help - stampa un messaggio di aiuto ed esce

-x, --expiration TEXT - setta il tempo in

secondi prima che il contratto finisca, di default è 30 secondi

-f, --force-unique - forza la transazione ad essere unica, questo consumerà banda extra e rimuoverà qualsiasi protezione contro errori multipli provenienti dalla stessa transazione

-s, --skip-sign - specifica se un wallet sbloccato può essere usato per firmare la transazione

-d, --dont-broadcast - non trasmette la transazione alla rete (la stampa al stdout)

-r, --ref-block TEXT - setta i riferimenti del numero del blocco o l'id del blocco usato per la TAPOS

-p, --permission TEXT - un account ed un livello di permesso, come in

account@permission (default è
account@active)

--max-cpu-usage-ms UINT - setta un
tetto massimo di millisecondi per
l'utilizzo della cpu, per l'esecuzione del
contratto (default è 0 cioè senza limiti)

--max-net-usage UINT - setta un tetto
massimo di net da usare, in bytes, per la
transazione (default è 0 cioè non ci sono
limiti)

Usage:

esempio

```
cleos system rex sellrex accountname1  
"1 REX"
```

Costituzione di EOS

Una delle cose che alla community di EOS piace è la promessa di una governance sulla blockchain, infatti ogni smart contract, sulla blockchain di EOS, è accompagnato da un Contratto Ricardiano (che definisce i termini e le condizioni legali, come richiesto dall'articolo VII - Open Source, della costituzione ad interim):

"Each Member who makes available a smart contract on this blockchain shall be a Developer. Each Developer shall offer their smart contracts via a free and open source license, and each smart contract shall be documented with a

Ricardian Contract stating the intent of all parties and naming the Arbitration Forum that will resolve disputes arising from that contract."

Quindi in sostanza le persone interagivano con la blockchain di EOS senza che questi fossero d'accordo con la costituzione di EOS, rendendo di fatto la cosa peggiore di un link ai termini e condizioni (ToS - Terms of Conditions), portando in molti a credere che EOS non avesse una costituzione od un documento simile.

La costituzione è molto importante perché chiarifica cosa è lecito e cosa no definendo regole precise a cui i

partecipanti devono sottostare, i BP devono seguire questo Contratto Ricardiano per registrarsi come BP^[12]:

The intent of the `{{ regproducer }}` action is to register an account as a BP candidate.

I, `{{producer}}`, hereby nominate myself for consideration as an elected block producer.

If `{{producer}}` is selected to produce blocks by the eosio contract, I will sign blocks with `{{producer_key}}` and I hereby attest that I will keep this key secret and secure.

If `{{producer}}` is unable to perform obligations under this contract I will resign my position by resubmitting this

contract with the null producer key.

I acknowledge that a block is 'objectively valid' if it conforms to the deterministic blockchain rules in force at the time of its creation, and is 'objectively invalid' if it fails to conform to those rules.

{{producer}} hereby agrees to only use {{producer_key}} to sign messages under the following scenarios:
proposing an objectively valid block at the time appointed

by the block scheduling algorithm pre-confirming a block produced by another producer in the schedule when I find said block objectively valid confirming a block for which {{producer}} has

received $2/3+$ pre-confirmation messages from other producers

I hereby accept liability for any and all provable damages that result from my: signing two different block proposals with the same timestamp with `{{producer_key}}` signing two different block proposals with the same block number with `{{producer_key}}` signing any block proposal which builds off of an objectively invalid block signing a pre-confirmation for an objectively invalid block signing a confirmation for a block for which I do not possess pre-confirmation messages from $2/3+$ other producers

I hereby agree that double-signing for a timestamp or block number in concert

with 2 or more other producers shall automatically be deemed malicious and subject to a fine equal to the past year of compensation received and immediate disqualification from being a producer, and other damages. An exception may be made if {{producer}} can demonstrate that the double-signing occurred due to a bug in the reference software; the burden of proof is on {{producer}}.

I hereby agree not to interfere with the producer election process. I agree to process all producer election transactions that occur in blocks I create, to sign all objectively valid blocks I create that contain election transactions, and to sign all pre-confirmations and confirmations necessary to facilitate

transfer of control to the next set of producers as determined by the system contract.

I hereby acknowledge that 2/3+ other elected producers may vote to disqualify {{producer}} in the event {{producer}} is unable to produce blocks or is unable to be reached, according to criteria agreed to among producers.

If {{producer}} qualifies for and chooses to collect compensation due to votes received, {{producer}} will provide a public endpoint allowing at least 100 peers to maintain synchronization with the blockchain and/or submit transactions to be

included. {{producer}} shall maintain at least 1 validating node with full state and signature checking and shall report any objectively invalid blocks produced by the active block producers. Reporting shall be via a method to be agreed to among producers, said method and reports to be made public.

The community agrees to allow {{producer}} to authenticate peers as necessary to prevent abuse and denial of service attacks; however, {{producer}} agrees not to discriminate against non-abusive peers.

I agree to process transactions on a FIFO best-effort basis and to honestly bill transactions for measured execution time.

I {{producer}} agree not to manipulate the contents of blocks in order to derive profit from: the order in which transactions are included the hash of the block that is produced

I, {{producer}}, hereby agree to disclose and attest under penalty of perjury all ultimate beneficial owners of my company who own more than 10% and all direct shareholders.

I, {{producer}}, hereby agree to cooperate with other block producers to carry out our respective and mutual obligations under this agreement, including but not limited to maintaining network stability and a valid blockchain.

I, {{producer}}, agree to maintain a

website hosted at `{{url}}` which contains up-to-date information on all disclosures required by this contract.

I, `{{producer}}`, agree to set `{{location}}` such that `{{producer}}` is scheduled with minimal latency between my previous and next peer.

I, `{{producer}}`, agree to maintain time synchronization within 10 ms of global atomic clock time, using a method agreed to among producers.

I, `{{producer}}`, agree not to produce blocks before my scheduled time unless I have received all blocks produced by the prior producer.

I, `{{producer}}`, agree not to publish blocks with timestamps more than

500ms in the future unless the prior block is more than 75% full by either CPU or network bandwidth metrics.

I, `{{producer}}`, agree not to set the RAM supply to more RAM than my nodes contain and to resign if I am unable to provide the RAM approved by 2/3+ producers, as shown in the system parameters.

Inoltre queste regole servono anche per prendere decisioni, da parte dei BP, in merito alla blockchain come ad esempio congelare dei fondi e situazioni simili, ponendo i BP in una situazione di rischio, quindi dando voto negativo, a meno che tutti i soggetti coinvolti hanno accettato la costituzione tramite un

contratto volontario. L'esempio in questo caso era quello dell'ECAF (EOS Community Arbitration Forum) che permetteva di risolvere controversie sulla blockchain, come ad esempio congelare i fondi di un account (caso di furto da parte di un altro account malevolo) dimostrando, con abbastanza prove, ma l'ECAF poteva sempre esitare a procedere con l'azione dato che l'utente finale non aveva firmato un contratto che dimostrasse di seguire la costituzione di EOS.

Per ovviare a questo tipo di problematiche si può pensare, ad esempio, che ogni BP firmasse un messaggio sulla blockchain dove

espressamente afferma di seguire la costituzione di EOS nel blocco corrente ed in tutti quelli precedenti, oppure modificare il Ricardian Contract dove le varie azioni fanno riferimento alla costituzione di EOS.

Prima di vedere com'è stata modificata la "costituzione ad interim" che regolava il tutto, vediamo come si presentava la stessa e cosa comprendeva^[13]:

Article I - No Initiation of Violence

Members shall not initiate violence or the threat of violence against another Member. Lawful prosecution of crimes with the goal of preserving life, liberty and property does not constitute initiation of violence.

Article II - No Perjury

Members shall be liable for losses caused by false or misleading attestations and shall forfeit any profit gained thereby.

Article III - Rights

The Members grant the right of contract and of private property to each other, therefore no property shall change hands except with the consent of the owner, by a valid Arbitrator's order, or via community referendum. This Constitution creates no positive rights for or between any Members.

Article IV - No Vote Buying

No Member shall offer nor accept anything of value in exchange for a vote of any type, nor shall any Member unduly influence the vote of another.

Article V - No Fiduciary

No Member nor EOS token holder shall have fiduciary responsibility to support the value of the EOS token. The Members do not authorize anyone to hold assets, borrow, nor contract on behalf of EOS token holders collectively. This blockchain has no owners, managers or fiduciaries; therefore, no Member shall have beneficial interest in more than 10% of the EOS token supply.

Article VI - Restitution

Each Member agrees that penalties for breach of contract may include, but are not limited to, fines, loss of account, and other restitution.

Article VII - Open Source

Each Member who makes available a smart contract on this blockchain shall be a Developer. Each Developer shall offer their smart contracts via a free and open source license, and each smart contract shall be documented with a Ricardian Contract stating the intent of all parties and naming the Arbitration Forum that will resolve disputes arising from that contract.

Article VIII - Language

Multi-lingual contracts must specify one prevailing language in case of dispute and the author of any translation shall be liable for losses due to their false, misleading, or ambiguous attested translations.

Article IX - Dispute Resolution

All disputes arising out of or in connection with this Constitution shall be finally settled under the Rules of Dispute Resolution of the EOS Core Arbitration Forum by one or more arbitrators appointed in accordance with the said Rules.

Article X - Choice of Law

Choice of law for disputes shall be, in order of precedence, this Constitution and the Maxims of Equity.

Article XI - Amending

This Constitution and its subordinate documents shall not be amended except by a vote of the token holders with no less than 15% vote participation among tokens and no fewer than 10% more Yes than No votes, sustained for 30 continuous days within a 120 day period.

Article XII - Publishing

Members may only publish information to the Blockchain that is within their right to publish. Furthermore, Members voluntarily consent for all Members to permanently and irrevocably retain a copy, analyze, and distribute all broadcast transactions and derivative information.

Article XIII - Informed Consent

All service providers who produce tools to facilitate the construction and signing of transactions on behalf of other Members shall present to said other Members the full Ricardian contract terms of this Constitution and other referenced contracts. Service providers shall be liable for losses resulting from failure to disclose the full Ricardian contract terms to users.

Article XIV - Severability

If any part of this Constitution is declared unenforceable or invalid, the remainder will continue to be valid and enforceable.

Article XV - Termination of Agreement

A Member is automatically released from all revocable obligations under this Constitution 3 years after the last transaction signed by that Member is incorporated into the blockchain. After 3 years of inactivity an account may be put up for auction and the proceeds distributed to all Members according to the system contract provisions then in effect for such redistribution.

Article XVI - Developer Liability

Members agree to hold software developers harmless for unintentional mistakes made in the expression of contractual intent, whether or not said mistakes were due to actual or perceived negligence.

Article XVII - Consideration

All rights and obligations under this Constitution are mutual and reciprocal and of equally significant value and cost to all parties.

Article XVIII - Acceptance

A contract is deemed accepted when a member signs a transaction which incorporates a TAPOS proof of a block whose implied state incorporates an ABI of said contract and said transaction is incorporated into the blockchain.

Article XIX - Counterparts

This Constitution may be executed in any number of counterparts, each of which when executed and delivered shall constitute a duplicate original, but all counterparts together shall constitute a single agreement.

Article XX - Interim Constitution

This constitution is interim and is intended to remain in effect until a permanent constitution is written and ratified in a referendum.

Leggendo la precedente costituzione alcuni di voi avranno notato che per approvare la nuova costituzione c'erano dei parametri da rispettare come ricordato dall'articolo XI della costituzione ad interim, però il tutto è stato approvato anche se quei parametri non sono stati rispettati, infatti la proposta di [referendum](#), fatta il 7 febbraio 2019 con scadenza l'8 maggio 2019, ha raccolto solo oltre 17 milioni di voti per un totale di 1,7 % del totale dei token, quindi non sufficienti, perchè

è richiesto almeno il 15% (più o meno oltre 75 milioni di voti sono necessari, mantenuti per oltre 30 giorni consecutivi), per cambiare il tutto, dato che deve essere scritta e ratificata tramite un referendum come previsto dall'articolo XX della costituzione ad interim.

Quindi com'è possibile che grazie solo all'1,7 % dei voti è stato possibile ratificare l'attuale costituzione ad interim, inoltre dato che la costituzione attuale è "ad interim" dovrebbe rappresentare un caso speciale e quindi permettere l'avvio di una "vera" costituzione? Infatti le condizioni dettate dalla costituzione ad interim erano troppo ottimistiche, perché si è

dimostrato come non sia sufficiente anche perché poco più del 6 % del totale degli account vota attivamente sulla blockchain rendendo di conseguenza molto difficile raggiungere quei valori.

Tutto questo è stato reso possibile grazie ad un altro articolo della costituzione ad interim, il XIV:

"If any part of this Constitution is declared unenforceable or invalid, the remainder will continue to be valid and enforceable."

Dove approvando la proposta dell'EUA da parte dei BP, di fatto ha reso invalido l'articolo XI, permettendo questi cambiamenti ed uno degli aspetti positivi della DPoS (Delegate Proof-of-

Stake) è che non richiedono un fork del token e non ci può essere un caso "EOS Classic", quindi tramite lo stake dei token che fanno i singoli e che delegano questi, votando, per supportare sia le varie proposte sia principalmente i Block Producer, che hanno permesso di approvare il tutto senza che si sia creata confusione o che attori prendessero strade diverse nel processo, avendo opinioni contrastanti.

Il giorno 12 aprile 2019 la costituzione di EOS fu cambiata passando dalla costituzione ad interim a

nuovo EUA (EOS User Agreement), che fu proposta dal BP EOS New York tramite un referendum^[14].

Si è scelto di chiamarla EUA (EOS User Agreement) invece di costituzione proprio perché il termine "costituzione" può essere percepito in maniera differente sia dalle differenze culturali che da quelle linguistiche, inoltre questo documento può essere cambiato, emendato e modificato in maniera diversa e migliore in futuro, perché è una specie di framework di "documenti di governo" permettendo di fare modifiche in maniera veloce e semplice.

All'interno dell'EUA possiamo trovare alcune azioni che determinano come deve essere sviluppato il tutto:

chain:id - è l'ID
aca376f206b8fc25a6ed44dbdc66547c36
eosio.prods - un account EOS con una
struttura di permessi dinamica che può
assumere privilegi dell'account eosio,
quando 15/21 dei BP sono d'accordo.

A seguire l'EUA:

Article I — User Acknowledgement of Risks

If User loses access to their EOS account on chain_id and has not taken appropriate measures to secure access to their EOS account by other means, the User acknowledges and agrees that that EOS account will become inaccessible. Users acknowledge that the

User has an adequate understanding of the risks, usage and intricacies of cryptographic tokens and blockchain-based software. The User acknowledges and agrees that the User is using the EOS blockchain at their sole risk.

Article II — Special User Types

Users who call regproducer agree to, and are bound by, the *regproducer* Ricardian Contract.

Article III — Consent of the EUA

The nature of the *EOS User Agreement* is such that it serves as a description of the current EOS Mainnet governance functions that are in place. These functions, enforced by code, do not require the consent of Users as these functions are inherent and systemic to the EOS Mainnet itself.

Article IV — Governing Documents

Any modifications to the *EUA* and *governing documents* may be made by eosio.prods. It is

admonished that a statement be crafted and issued through eosio.prods via eosio.forum referendum contract describing such a modification in advance.

Article V — Native Unit of Value

The native unit of value on EOS chain_id shall be the EOS token as defined and created by the eosio.token smart contract.

Article VI — Maintaining the EOS blockchain

eosio.prods will maintain the active blockchain codebase which includes, but is not limited to, the implementation of all modifications of all features, optimizations, and upgrades: present and

future.

Article VII — Network Funds

It is admonished that any altering of the state of any tokens contained within network fund accounts, or altering any pre-existing code that directly or indirectly governs the allocation, fulfillment, or distribution of any *network funds* be preceded by a statement crafted and issued by eosio.prods to the *eosio.forum* referendum system contract describing the effect in advance.

Article VIII — Freedom of Account Creation

Any current or future User is able to create an EOS Account without the

permission by any other User. eosio.prods may never affect an EOS User Account(s) without valid permission(s) which have been shared with eosio.prods by an EOS account. eosio.prods may charge a fee for any actions that are requested by other Users pertaining to an EOS account where permissions are shared.

Article IX — No Fiduciary

No User shall have a fiduciary purpose to support the value of the EOS token. No User can authorize anyone to hold assets, borrow, speak, contract on behalf of other EOS Users or the EOS blockchain chain_id collectively. This EOS blockchain shall have no owners, managers, or fiduciaries.

Article X — User Security

All items pertaining to personal account security, including but not limited to the safekeeping of private keys, is solely the responsibility of the User to secure.

Article XI — eosio.prods Limited Liability

The User acknowledges and agrees that, to the fullest extent permitted by any applicable law, this disclaimer of liability applies to any and all damages or injury whatsoever caused by or related to risks of, use of, or inability to use, the EOS token or the EOS blockchain chain_id under any cause of action whatsoever of any kind in any jurisdiction, including, without

limitation, actions for breach of warranty, breach of contract or tort (including negligence) and that eosio.prods, nor the individual permissions that operate it, shall not be liable for any indirect, incidental, special, exemplary or consequential damages, including for loss of profits, goodwill or data.

Gli Smart Contract

Uno smart contract o contratto intelligente in italiano è un codice software progettato come un contratto per l'auto-applicazione automatizzata, il che significa che avvia determinate azioni dopo che sono state soddisfatte condizioni predeterminate. Gli smart contract possono essere utilizzati, ad esempio, come accordi digitali che intermediano lo scambio di cripto monete (o qualsiasi altra risorsa digitale) tra due parti.

Una volta stabiliti i termini dell'accordo, lo smart contract verifica

il loro adempimento e le attività sono distribuite in conformità allo stesso, dove di solito c'è una condizione "if... then" quindi se si verifica questo evento allora fai questo.

Ovviamente lo smart contract può essere applicato in tutti i settori e non c'è veramente un limite in questo dato che le informazioni che vengono scritte al suo interno sono molto flessibili e variano in base allo scopo finale che dovranno fare, permettendo in pratica di coprire qualsiasi aspetto che coinvolga una o più condizioni.

Anche nell'ordinamento italiano è stato stabilito lo smart contract e riconosciuto come valore legale e quindi a tutti gli effetti equiparato ad un

contratto tradizionale, il tutto è possibile reperire nella Legge 11 febbraio 2019, n.12 e precisamente l'Art. 8-ter comma 2, dove dispone:

“Si definisce smart contract un programma per elaboratore che opera su tecnologie basate su registri distribuiti e la cui esecuzione vincola automaticamente due o più parti sulla base di effetti predefiniti dalle stesse. Gli smart contract soddisfano il requisito della forma scritta previa identificazione informatica delle parti interessate, attraverso un processo avente i requisiti fissati dall’Agenzia per l’Italia digitale con linee guida da adottare entro novanta giorni dalla data

di entrata in vigore della legge di conversione del presente decreto”.

Da ricordare, inoltre, che siccome può essere utilizzato come un contratto tradizionale, è meglio avere anche un supporto legale per la sua stesura, dato che per la sua verifica ed attivazione non è necessario tale supporto, ma solo uno sviluppatore che realizzerà lo smart contract con i parametri e osservazioni della figura legale per avere un contratto preciso ed efficace sotto diversi aspetti, quindi in futuro, opinione personale, si creeranno professioni dedicate a questo nuovo tipo di settore come ad esempio l'avvocato blockchain e lo sviluppatore di smart contract definendo 2 categorie dedite solo a quelle funzioni, anche se al

momento non ci sono percorsi ufficiali e riconosciuti dallo Stato sotto quel punto di vista ed i diversi "esperti" hanno seguito diversi master dedicati per approfondire l'argomento.

Smart Contract nella blockchain di EOS

Per realizzare degli smart contract nella blockchain di EOS, bisogna conoscere il linguaggio di programmazione C++ che è il linguaggio utilizzato principalmente per operare con questa blockchain, ma ovviamente non è esclusivo dato che ci sono alcuni strumenti ed altri linguaggi che si possono utilizzare, ma in questo

caso noi faremo riferimento al linguaggio di programmazione C++, quindi tutti gli esempi sono da interpretare con quel linguaggio.

Installare il CDT

Prima di tutto dobbiamo installare il CDT (EOSIO contract Development Toolkit), che è un insieme di strumenti relativi alla compilazione dei contratti; il CDT supporta il MacOS X brew, Linux Debian e i pacchetti RPM, ed in base alle vostre preferenze potete installare la versione che preferite.

Homebrew (MacOS X)

installazione:

```
brew tap eosio/eosio.cdt
```

```
brew install eosio.cdt
```

disinstallazione:

```
brew remove eosio.cdt
```

Ubuntu (Debian)

Installazione:

wget

https://github.com/EOSIO/eosio.cdt/releases/download/v1.6.1/eosio.cdt_1.6.1-1_amd64.deb

```
sudo apt install ./eosio.cdt_1.6.1-1_amd64.deb
```

disinstallazione:

```
sudo apt remove eosio.cdt
```

CentOS/Redhat (RPM)

Installazione:

wget

https://github.com/EOSIO/eosio.cdt/releases/download/v1.6.1/eosio.cdt_1.6.1-1.centos-x86_64.rpm

```
sudo yum install ./eosio.cdt-1.6.1-1.centos-x86_64.rpm
```

disinstallazione:

```
$ sudo yum remove eosio.cdt
```

Installare dai sorgenti

Il percorso dove eosio.cdt sarà clonato non è importante dato che sarà installato eosio.cdt come binario nei passaggi finali, quindi possiamo anche salvare eosio.cdt nella directory contract.

```
cd CONTRACTS_DIR
```


scaricare la versione relativa, in questo caso la 1.6.1 della repository eosio.cdt, che impiegherà circa 30 minuti il processo:

```
git clone --recursive
https://github.com/eosio/eosio.cdt --
branch v1.6.1 --single-branch
cd eosio.cdt
```

poi i passaggi successivi sono build e install:

```
./build.sh
sudo ./install.sh
```

il comando precedente deve essere avviato con sudo perchè eosio.cdt installerà diversi codici binari

localmente e quindi ci verrà richiesta anche la password dell'account del pc, ed inoltre potremmo accedere da ovunque questi codici. Nel caso ci fossero errori possiamo cercarli nella stringa `/usr/local/include/eosiolib/` se troviamo `rm` `-fr` `/usr/local/include/eosiolib/` o navigando in `/usr/local/include/` dobbiamo cancellare eosiolib.

Hello World!

Hello, world! (in [italiano](#) "Ciao, mondo!") è un semplice programma informatico che produce come risultato - output - la scritta "Hello, world!" o altre varianti. Il programma non fa nient'altro che far comparire a schermo questa

scritta e, per tradizione, è doveroso citarlo e farne un esempio con la blockchain di EOS. Di solito uno smart contract richiede 2 file l'header che è un file .hpp dove è usato per dichiarare tutte le variabili pubbliche e i metodi del contratto che saranno accessi dall'esterno, e poi il file principale .cpp. Prima di tutto dobbiamo creare una cartella chiamata "hello" nella directory che avremmo creato in precedenza in questo caso la nostra sarà "CONTRACT_DIR":

```
cd CONTRACTS_DIR
mkdir hello
cd hello
```

Poi dobbiamo creare un nuovo file "hello.cpp" ed aprirlo con un editor che preferiamo:

```
touch hello.cpp
```

A questo punto dobbiamo recarci in C++, dove dobbiamo inserire eosio.hpp all'inizio per includere i file relativi per scrivere lo smart contract:

```
#include <eosio/eosio.hpp>
```

Poi possiamo usare eosio namespace per ridurre l'ingombro del codice e quindi invece di scrivere per esempio eosio::print("foo"), possiamo scrivere semplicemente print("foo"):

```
using namespace eosio;
```

Dopo di che dobbiamo creare una classe C++11 standard, i contract class hanno bisogno di estendere le classi eosio::contract che sono incluse nell'intestazione eosio.hpp:

```
#include <eosio/eosio.hpp>
```

```
using namespace eosio;
```

```
class [[eosio::contract]] hello : public  
contract {};
```

Adesso abbiamo un contratto vuoto e non molto utile, quindi dobbiamo

aggiungere uno specifico accesso pubblico ed usare la dichiarazione using, che permette di scrivere un codice più conciso:

```
#include <eosio/eosio.hpp>
```

```
using namespace eosio;
```

```
class [[eosio::contract]] hello : public  
contract {  
    public:  
        using contract::contract;  
};
```

Ora il contratto ha bisogno di fare qualcosa ed in questo caso dato che vogliamo far mostrare la scritta "hello

world" dobbiamo aggiungere le azioni che accettano i parametri "name" e stampa il risultato nell'output:

```
#include <eosio/eosio.hpp>
```

```
using namespace eosio;
```

```
class [[eosio::contract]] hello : public  
contract {  
    public:  
        using contract::contract;  
  
        [[eosio::action]]  
        void hi( name user ) {  
            print( "Hello, ", user);  
        }  
};
```

L'azione precedente accetta il parametri chiamato `user` che è un `type name`, infatti `EOSIO` comprende un numero di `typedefs` (type definitions), e uno dei più comuni è `proper_name`, poi usando le librerie precedentemente incluse, possiamo usare `eosio::print`, possiamo concatenare la stringa e stampare il parametro `user` ed utilizziamo l'inizializzazione forzata di `name{user}` per rendere stampabile il parametro `user`.

Da notare che il generatore `ABI GLOSSARY:ABI` in `eosio.cdt` non conosce l'azione `hi()` senza un attributo e quindi dobbiamo aggiungere l'attributo stile `C++11` sopra l'azione così da avere

un output più affidabile:

```
#include <eosio/eosio.hpp>
```

```
using namespace eosio;
```

```
class [[eosio::contract]] hello : public  
contract {  
    public:  
        using contract::contract;  
  
        [[eosio::action]]  
        void hi( name user ) {  
            print( "Hello, ", user);  
        }  
};
```

Ovviamente possiamo anche compilare il codice in web assembly (.wasm) come segue:

```
eosio-cpp hello.cpp -o hello.wasm
```

Quando si crea un contratto, questo è creato su un account e l'account diventa l'interfaccia del contratto ed ovviamente dobbiamo utilizzare le chiavi pubbliche:

```
cleos wallet keys
```

Poi dobbiamo creare un account per il contratto utilizzando il comando `cleos create account`:

```
cleos create account eosio hello  
YOUR_PUBLIC_KEY -p eosio@active
```

Poi dobbiamo compilare il contratto wasm con il comando `cleos set contract`, da notare che dobbiamo avere il percorso esatto della cartella dove si trova il relativo file:

```
cleos set contract hello  
CONTRACTS_DIR/hello -p  
hello@active
```

Ora che il contratto è settato dobbiamo inserire le azioni:

```
cleos push action hello hi ['"bob"'] -p  
bob@active
```

che darà come output questo:

```
executed                               transaction:
4c10c1426c16b1656e802f33026775947.
244 bytes 1000 cycles
#             hello.code             <=
hello.code::hi      {"user":"bob"}
>> Hello, bob
```

Inoltre come è stato scritto il contratto permette questa operazione di salutare qualsiasi utente:

```
cleos push action hello hi ["bob"] -p
alice@active
```

che darà come output questo:

```
executed                                transaction:
28d92256c8ffd8b0255be324e4596b7c74
244 bytes 1000 cycles
#                                hello.code    <=
hello.code::hi                    {"user":"bob"}
>> Hello, bob
```

In questo caso "alice" è quella che ha autorizzato l'operazione e user è solo un argomento, mentre se modifichiamo il contratto, autorizzando "alice" ad essere lo stesso utente che risponde al saluto, ed utilizziamo il metodo `require_auth`, dove si prende il parametro `name` e si fa un check per vedere se i parametri corrispondono.

In C++ è:

```
void hi( name user ) {  
    require_auth( user );  
    print( "Hello, ", name {user} );  
}
```

Adesso dobbiamo ricompilare il contratto:

```
eosio-cpp -abigen -o hello.wasm  
hello.cpp
```

ed aggiornarlo:

```
cleos set contract hello  
CONTRACTS_DIR/hello -p  
hello@active
```

Adesso eseguiamo l'azione con I parametri diversi:

```
cleos push action hello hi ["bob"] -p  
alice@active
```

In questo modo il parametro `require_auth` fermerà la transazione e darà un messaggio di errore:

```
Error 3090004: Missing required  
authority
```

Ensure that you have the related authority inside your transaction!;

If you are currently using 'cleos push action' command, try to add the relevant authority using -p option.

Ora con le modifiche che abbiamo fatto, il contratto verifica che il nome fornito in name user sia lo stesso dell'utente autorizzato, ed infatti se mettiamo i valori corretti questo sarà il risultato:

```
cleos push action hello hi ["alice"] -p
alice@active
executed                               transaction:
235bd766c2097f4a698cfb948eb2e70953
244 bytes 1000 cycles
#      hello  <=  hello::hi
{"user":"alice"}
>> Hello, alice
```

Sicurezza negli smart contract

Ovviamente quando si crea uno smart contract c'è sempre che qualcosa vada male e che una volta inizializzato ed implementato possa essere sfruttato per qualcosa di diverso o per azioni fraudolente utilizzando lo stesso smart contract perchè non è stato controllato e certificato il relativo codice.

Una cosa utile che si può fare quando si finisce di scrivere uno smart contract è quello di farlo certificare/controllare da un soggetto terzo affinché lo si testi in maniera pesante ed evitare che questo una volta online possa presentare delle

falle che possono essere fruttate in malo modo.

Durante la creazione di uno smart contract per la blockchain di EOS è meglio prendere alcuni accorgimenti e pratiche utili come ad esempio:

- far fare un controllo sul codice da un'azienda esterna per prima di lanciare il tutto sulla mainnet;
- fare il necessario debugging del codice prima di rilasciare il tutto sulla testnet;
- settare dei tassi di limiti di trasferimenti e di prelievi per evitare un'eccessiva perdita durante il lancio iniziale dello smart contract e predisporre anche un piano di caccia ai bug per ricompensare coloro che contribuiscono a trovare le relative falle

nel codice;

- utilizzare dei killswitch per congelare il contratto quando viene rilevato un bug;

- informare gli utenti sulle vulnerabilità risolte

- avere un codice open source così da permettere anche agli sviluppatori indipendenti la possibilità di implementare il codice, migliorarlo e trovare eventuali errori in maniera rapida ed efficace.

Comprendere i file ABI

L'ABI (Application Binary Interface) è un file basato json che converte le azioni

dell'utente nella loro rappresentazione json e codice binario, inoltre descrive anche come convertire lo stato del database da e per il json, poi una volta che abbiamo descritto il nostro contratto con l'ABI gli sviluppatori e gli utenti potranno interagire tramite json.

I file ABI possono essere generati usando l'utility eosio-cpp fornita da eosio.cdt, comunque ci sono diverse situazioni che la generazione dell'ABI causi malfunzionamenti o che fallisca completamente, e quindi è meglio saper comprendere com i file ABI lavorino per avere una maggiore comprensione di come fare debug.

Creare un file ABI

Iniziamo creando un file ABI vuoto che lo chiameremo eosio.token.abi

```
{  
  "version": "eosio::abi/1.0",  
  "types": [],  
  "structs": [],  
  "actions": [],  
  "tables": [],  
  "ricardian_clauses": [],  
  "abi_extensions": [],  
  "___comment" : ""  
}
```

types

Un ABI abilita qualsiasi client od interfaccia ad interpretare e generare una GUI per il contratto, per questo bisogna lavorare in una certa maniera, descrivendo types personalizzate che sono usati come parametri nelle azioni pubbliche o le strutture che devono essere descritte nell'ABI:

```
{  
  "new_type_name": "name",  
  "type": "name"
```

```
}
```

ottenendo un risultato simile:

```
{
```

```
  "version": "eosio::abi/1.0",
```

```
  "types": [{
```

```
    "new_type_name": "name",
```

```
    "type": "name"
```

```
  }],
```

```
  "structs": [],
```

```
  "actions": [],
```

```
  "tables": [],
```

```
  "ricardian_clauses": [],
```

```
  "abi_extensions": []
```

```
}
```

struct

Anche le struct devono essere descritte nell'ABI, dove una definizione di un oggetto struct in json assomiglia a questo:

```
{
  "name": "issue", //The name
  "base": "",      //Inheritance,
parent struct
  "fields": []     //Array of field
objects describing the struct's fields.
}
```


fields

In un contratto ci possono essere un numero di struct che è necessario definire (per esempio può essere il tipo di asset o la `maximum_supply` di un token) ma che non tutte le struct devono essere definite dato che alcune corrispondono a delle azioni dei parametri, e queste possono essere implicit structs ed explicit structs.

actions

Un'azione dell'oggetto definito nel json può essere la seguente, dove si descrive l'azione di un contratto `eosio.token`,

aggregando tutte le funzioni pubbliche descritte nel file header:

```
{  
  "name": "transfer",           //The name  
of the action as defined in the contract  
  "type": "transfer",         //The name  
of the implicit struct as described in the  
ABI  
  "ricardian_contract": ""    //An  
optional ricardian clause to associate to  
this action describing its intended  
functionality.  
}
```

tables

come possiamo immaginare, describe una tabella:

```
{  
  "name": "", //The name of the table,  
determined during instantiation.  
  "type": "", //The table's  
corresponding struct  
  "index_type": "", //The type of primary  
index of this table  
  "key_names" : [], //An array of key  
names, length must equal length of  
key_types member  
  "key_types" : [] //An array of key  
types that correspond to key names array  
member, length of array must equal  
length of key names array.
```

```
}
```

un ABI relativo ad un contratto eosio.token avrà le seguenti caratteristiche:

```
{  
  "version": "eosio::abi/1.0",  
  "types": [  
    {  
      "new_type_name": "name",  
      "type": "name"  
    }  
  ],  
  "structs": [  
    {  
      "name": "create",  
      "base": "",  
    }  
  ]  
}
```

```
"fields": [  
  {  
    "name": "issuer",  
    "type": "name"  
  },  
  {  
    "name": "maximum_supply",  
    "type": "asset"  
  }  
],  
{  
  "name": "issue",  
  "base": "",  
  "fields": [  
    {  
      "name": "to",  
      "type": "name"
```

```
    },  
    {  
      "name": "quantity",  
      "type": "asset"  
    },  
    {  
      "name": "memo",  
      "type": "string"  
    }  
  ]  
},  
{  
  "name": "retire",  
  "base": "",  
  "fields": [  
    {  
      "name": "quantity",  
      "type": "asset"
```

```
    },  
    {  
      "name": "memo",  
      "type": "string"  
    }  
  ]  
},  
{  
  "name": "close",  
  "base": "",  
  "fields": [  
    {  
      "name": "owner",  
      "type": "name"  
    },  
    {  
      "name": "symbol",  
      "type": "symbol"  
    }  
  ]  
}
```

```
    }  
  ]  
},  
{  
  "name": "transfer",  
  "base": "",  
  "fields": [  
    {  
      "name": "from",  
      "type": "name"  
    },  
    {  
      "name": "to",  
      "type": "name"  
    },  
    {  
      "name": "quantity",  
      "type": "asset"  
    }  
  ]  
}
```



```
    },  
    {  
      "name": "memo",  
      "type": "string"  
    }  
  ]  
},  
{  
  "name": "account",  
  "base": "",  
  "fields": [  
    {  
      "name": "balance",  
      "type": "asset"  
    }  
  ]  
},  
{
```

```
"name": "currency_stats",  
"base": "",  
"fields": [  
  {  
    "name": "supply",  
    "type": "asset"  
  },  
  {  
    "name": "max_supply",  
    "type": "asset"  
  },  
  {  
    "name": "issuer",  
    "type": "name"  
  }  
],  
}
```

```
"actions": [  
  {  
    "name": "transfer",  
    "type": "transfer",  
    "ricardian_contract": ""  
  },  
  {  
    "name": "issue",  
    "type": "issue",  
    "ricardian_contract": ""  
  },  
  {  
    "name": "retire",  
    "type": "retire",  
    "ricardian_contract": ""  
  },  
  {  
    "name": "create",
```

```
"type": "create",
"ricardian_contract": ""
},
{
"name": "close",
"type": "close",
"ricardian_contract": ""
}
],
"tables": [
{
"name": "accounts",
"type": "account",
"index_type": "i64",
"key_names": ["currency"],
"key_types": ["uint64"]
},
{
```

```
"name": "stat",
"type": "currency_stats",
"index_type": "i64",
"key_names" : ["currency"],
"key_types" : ["uint64"]
}
],
"ricardian_clauses": [],
"abi_extensions": []
}
```

vectors

Se dobbiamo descrivere un vettore nel file ABI basta aggiungere [] e quindi se dobbiamo descrivere, ad esempio, un vettore per il livello di permesso allora

lo faremo in questo modo:
permission_level[]

struct base

Anche se raro si può usare una struct ABI base che fa riferimento ad altre istanze di struct che sono descritte nello stesso file ABI.

ricardian clauses

Descrivono un particolare intento di un'azione determinata dove può essere utilizzato anche per stabilire termini tra in mandante ed il contratto.

ABI extensions

Serve in futuro per permettere a vecchi client di utilizzare e skippare ai nuovi client e quindi interpretare meglio il codice.

maintenance

Ogni volta che si cambia struct, si aggiunge una tabella, oppure un'azione od un parametro dell'azione o utilizzare un nuovo tipo, è necessario di ricrodare di aggiornare il file ABI.

Capitolo 4

nodeos, cleos e keosd

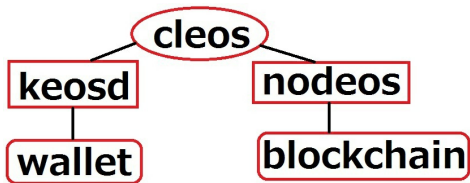
Il software EOSIO comprende una serie di programmi per la gestione di vari aspetti della blockchain, questi sono:

- nodeos (node + eos = nodeos), è il cuore dei nodi demoni di EOSIO che possono essere configurati con dei plugin per far girare un nodo, alcuni esempi sono quelli di produrre blocchi, endpoint API dedicate, sviluppare in maniera locale;

- cleos (cli + eos = cleos), è l'interfaccia

a linea di comando per interagire con la blockchain e gestire i wallet;

- keosd (key + eos = keosd), è un componente che mette al sicuro le chiavi EOSIO nei wallet.



Prima di partire nello specifico dobbiamo installare eosio in base al nostro sistema operativo:

MacOS X Brew

```
brew tap eosio/eosio
```

```
brew install eosio
```

Ubuntu 18.04 Debian Package Install

```
wget
```

```
https://github.com/EOSIO/eos/releases/download/eosio\_1.7.0-1-ubuntu-18.04\_amd64.deb
```

```
sudo apt install ./eosio_1.7.0-1-ubuntu-18.04_amd64.deb
```

Ubuntu 16.04 Debian Package Install

```
wget
```

```
https://github.com/EOSIO/eos/releases/d  
1-ubuntu-16.04_amd64.deb  
sudo apt install ./eosio_1.7.0-1-ubuntu-  
16.04_amd64.deb
```

CentOS RPM Package Install

```
wget  
https://github.com/EOSIO/eos/releases/d  
1.7.0-1.el7.x86_64.rpm  
sudo yum install ./eosio-1.7.0-  
1.el7.x86_64.rpm
```

Fedora RPM Package Install

```
wget
```

```
https://github.com/EOSIO/eos/releases/download/1.7.0-1.fc27.x86\_64.rpm  
sudo yum install ./eosio-1.7.0-1.fc27.x86_64.rpm
```

Poi dobbiamo settare la directory, che in questo caso sarà contracts:

```
mkdir contracts
```

```
cd contracts
```

Infine dobbiamo inserire il nostro percorso usando il comando pwd

Guida nodeos

Prerequisito

Per prima cosa bisogna installare il docker, reperibile al seguente indirizzo:

<https://www.docker.com/community-edition>

Primo passo - prendere l'immagine del software

Il passo successivo è prendere l'immagine che contiene il software compilato, che è un'immagine Ubuntu:

```
docker pull eosio/eos-dev:v1.5.2
```

Secondo passo - creare il network

Utilizzeremo il comando `network` del `docker` per creare una rete per `nodeos` e `keosd` da condividere:

```
docker network create eosdev
```

Terzo passo - boot container

nodeos (core daemon)


```
$ docker run \  
--name nodeeos -d -p 8888:8888 \  
--network eosdev \  
-v /tmp/eosio/work:/work \  
-v /tmp/eosio/data:/mnt/dev/data \  
-v /tmp/eosio/config:/mnt/dev/config \  
eosio/eos-dev \  
/bin/bash -c \  
"nodeos -e -p eosio \  
--plugin eosio::producer_plugin \  
--plugin eosio::history_plugin \  
--plugin eosio::chain_api_plugin \  
--plugin eosio::history_api_plugin \  
--plugin eosio::http_plugin \  
-d /mnt/dev/data \  
--config-dir /mnt/dev/config \  
--http-server-address=0.0.0.0:8888 \  
"
```

```
--access-control-allow-origin=* \  
--contracts-console \  
--http-validate-host=false"
```

Questi settaggi portano a questi risultati:

- il forward verso la porta 8888;
- la connessione a eosdev della rete locale che abbiamo creato in precedenza;
- creazione di 3 alias sul drive locale per il contenitore del docker;
- avvia l'esecuzione di nodeos in bash, questo comando carica tutti i plugin di base, setta gli indirizzi del server, abilita il CORS e aggiunge qualche contratto per il debug;
- monta alcune nella directory in /tmp

sulla macchina locale, e cambiando queste cartelle, impatteranno su tutto il sistema del contenitore del docker.

avvio di keosd (wallet e keystore)

```
docker run -d --name keosd --  
network=eosdev \  
-i eosio/eos-dev /bin/bash -c "keosd --  
http-server-address=0.0.0.0:9876"
```

Quarto passo - controllare l'installazione

- controllare che nodeos produce

blocchi

digitiamo il seguente comando:

```
docker logs --tail 10 nodeos
```

Avendo come risultato un output del genere sulla console:

```
1929001ms                               thread-0  
producer_plugin.cpp:585  
block_production_loop ] Produced block  
0000366974ce4e2a... #13929 @ 2018-  
05-23T16:32:09.000 signed by eosio  
[txs: 0, lib: 13928, confirmed: 0]  
1929502ms                               thread-0  
producer_plugin.cpp:585  
block_production_loop ] Produced block
```

0000366aea085023... #13930 @ 2018-05-23T16:32:09.500 signed by eosio [trxs: 0, lib: 13929, confirmed: 0] 1930002ms thread-0
producer_plugin.cpp:585
block_production_loop] Produced block
0000366b7f074fdd... #13931 @ 2018-05-23T16:32:10.000 signed by eosio [trxs: 0, lib: 13930, confirmed: 0] 1930501ms thread-0
producer_plugin.cpp:585
block_production_loop] Produced block
0000366cd8222adb... #13932 @ 2018-05-23T16:32:10.500 signed by eosio [trxs: 0, lib: 13931, confirmed: 0] 1931002ms thread-0
producer_plugin.cpp:585
block_production_loop] Produced block

0000366d5c1ec38d... #13933 @ 2018-05-23T16:32:11.000 signed by eosio [trxs: 0, lib: 13932, confirmed: 0] 1931501ms thread-0
producer_plugin.cpp:585
block_production_loop] Produced block
0000366e45c1f235... #13934 @ 2018-05-23T16:32:11.500 signed by eosio [trxs: 0, lib: 13933, confirmed: 0] 1932001ms thread-0
producer_plugin.cpp:585
block_production_loop] Produced block
0000366f98adb324... #13935 @ 2018-05-23T16:32:12.000 signed by eosio [trxs: 0, lib: 13934, confirmed: 0] 1932501ms thread-0
producer_plugin.cpp:585
block_production_loop] Produced block

```
00003670a0f01daa... #13936 @ 2018-05-23T16:32:12.500 signed by eosio [txs: 0, lib: 13935, confirmed: 0] 1933001ms thread-0 producer_plugin.cpp:585 block_production_loop ] Produced block 00003671e8b36e1e... #13937 @ 2018-05-23T16:32:13.000 signed by eosio [txs: 0, lib: 13936, confirmed: 0] 1933501ms thread-0 producer_plugin.cpp:585 block_production_loop ] Produced block 0000367257fe1623... #13938 @ 2018-05-23T16:32:13.500 signed by eosio [txs: 0, lib: 13937, confirmed: 0]
```

Poi andando oltre e bisogna uscire dal bash tramite ctrl-c oppure digitare exit e

premere invio.

- controllare il wallet

apriamo la shell nodeos:

```
docker exec -it keosd bash
```

poi bisogna digitare il seguente comando:

```
cleos --wallet-url http://127.0.0.1:9876  
wallet list keys
```


e dobbiamo visualizzare il seguente messaggio:

Wallets:

```
[]
```

Adesso abbiamo la certezza che keosd sta girando correttamente e quindi digitiamo exit e premiamo invio per uscire dalla shell di keosd, da questo punto in avanti non c'è bisogno di entrare il container con il bash e potremmo eseguire i comandi dal nostro sistema locale (Linux o Mac).

- controllare gli endpoint nodeos

Questo permetterà di controllare che gli RPC API lavorano correttamente, prendendone uno.

Controllare l'endpoint `get_info` fornito da `chain_api_plugin` nel proprio browser:

```
http://localhost:8888/v1/chain/get_info
```

lo stesso controllo lo dobbiamo fare nella console

```
curl
```

```
http://localhost:8888/v1/chain/get_info
```

Quinto passo - alias Cleos

Dato che non vogliamo entrare nel container bash del docker ogni volta che interagiamo con nodeos o keosd, andiamo oltre e rendiamo cleos più semplice da usare.

Per prima cosa dobbiamo trovare l'indirizzo IP di keosd, digitando il seguente comando:

```
docker network inspect eosdev
```

e dovremmo vedere una cosa del genere e trovare l'indirizzo IP di keosd:

```
[  
  {  
    "Name": "eosdev",  
    "Id":
```

```
"b24a6ae0b8a559212a1bf94cac77a1774
    "Created": "2018-09-
05T01:20:26.4181748Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
        "Driver": "default",
        "Options": {},
        "Config": [
            {
                "Subnet": "172.18.0.0/16",
                "Gateway": "172.18.0.1"
            }
        ]
    },
    "Internal": false,
    "Attachable": false,
```

```
"Ingress": false,  
"ConfigFrom": {  
  "Network": ""  
},  
"ConfigOnly": false,  
"Containers": {
```

```
"2b6d8421243f8b02d19caf84735f05118  
{
```

```
  "Name": "nodeos",  
  "EndpointID":  
"164d47d6f79b4b7348154485a6c79d97",  
  "MacAddress":  
"02:42:ac:12:00:02",  
  "IPv4Address":  
"172.18.0.2/16",  
  "IPv6Address": ""  
},
```

```
"ffe5f373dffdf9b31b69394416cadb7e29":  
{  
    "Name": "keosd",  
    "EndpointID":  
"ac8c6bf49ba698c226631b41d98cb49f7",  
    "MacAddress":  
"02:42:ac:12:00:03",  
    "IPv4Address":  
"172.18.0.3/16",  
    "IPv6Address": ""  
}  
,  
"Options": {},  
"Labels": {}  
}  
]
```

Poi con il seguente comando creeremo un alias per cleos sulla nostra macchina e collegare quell'alias ad un comando che sarà eseguito all'interno del container nodeos, ecco perché i parametri `--url` fanno riferimento a localhost della porta 8888, mentre i parametri `--wallet-url` si riferisce a keosd:

```
alias cleos='docker exec -it nodeos  
/opt/eosio/bin/cleos --url  
http://127.0.0.1:8888 --wallet-url  
http://[keosd_ip]:9876'
```

Avviare nodeos

Nodeos può essere avviato dalla linea di comando ed il comportamento di nodeos è determinato da quale plugin stiamo utilizzando e la configurazione usata per ogni plugin, nodeos stesso ha alcune opzioni e queste permettono di settare la data directory, dove i dati della blockchain saranno salvati e il punto di configurazione per i plugin e per i logging.

ad esempio:

```
nodeos -e -p eosio --plugin
eosio::producer_plugin --plugin
eosio::chain_api_plugin --plugin
eosio::http_plugin --plugin
eosio::state_history_plugin --data-dir
```



```
/Users/mydir/eosio/data      --config-dir
/Users/mydir/eosio/config    --access-
control-allow-origin='*'     --contracts-
console  --http-validate-host=false  --
state-history-dir /shpdata --trace-history
--chain-state-history        --verbose-http-
errors  --filter-on='*'     --disable-replay-
opts >> nodeos.log 2>&1 &
```

Configurazione nodeos

Nodeos può essere configurato usando sia l'interfaccia a linea di comando (CLI - Command Line Interface) e sia tramite un file di configurazione, config.ini. Tutte le opzioni di CLI si possono trovare avviando \$ nodeos --help.

Ogni opzione CLI corrisponde ad un settaggio nel file config.ini, dove per esempio `--plugin eosio::chain_api_plugin` può essere fatto nello stesso modo ma aggiungendo `plugin = eosio::chain_api_plugin` a file config.ini.

Un file personalizzato di config.ini può essere usato ed eseguito tramite il seguente comando:

```
$ nodeos --config path/to/config.ini
```

Configurare il percorso

Il file config.ini può essere trovato nei seguenti percorsi:

- Mac OS: ~/Library/Application Support/eosio/nodeos/config

-

Linux: ~/.local/share/eosio/nodeos/config

Opzioni nodeos

Un esempio proveniente dall'output del comando `$ nodeos --help` lo troviamo di seguito al netto delle opzioni per i plugin:

Application Options:

config.ini

-c [--config] arg (=config.ini)

Configuration file name relative to
config-dir

-l [--logconf] arg (=logging.json)

Logging configuration file name/path
for library

users

Settare nodeos

Generalmente nodeos viene eseguito in 2 modi:

- producing node, che si connette alla

rete peer to peer ed attivamente produce i nuovi blocchi;

- non-producing node, che si connette alla rete peer to peer ma in questo modo non si producono blocchi ma si verificano semplicemente i blocchi e si ha una copia della blockchain, questa modalità serve per monitorare la blockchain.

Producing Node

Le seguenti istruzioni prevedono di lanciare un producing node sulla rete con un sistema di contratto caricato, quindi queste istruzioni non funzionano

su un nodo di sviluppo di default usando la funzione nativa, od uno senza il sistema di contratto caricato.

Registrare un account come producer

Affinché un account sia eleggibile come producer bisogna registrare l'account come producer, utilizzeremo cleos per settare il tutto.

cleos system reproducer

```
cleos system reproducer accountname1  
EOS1234534...      http://producer.site
```

Antarctica

Settare un nome per il producer

Bisogna settare l'opzione producer-name nel file config.ini ed aggiungere il nostro account:

```
# ID of producer controlled by this node  
(e.g. inita; may specify multiple times)  
(eosio::producer_plugin)  
producer-name = youraccount
```

Settare la firma del producer

Bisogna settare la chiave privata per il

producer, mentre la chiave pubblica dovrebbe avere l'autorità per il producer come definito precedentemente:

signature-provider is defined with a tuple

public-key - A valid EOSIO public key in form of a string.

provider-spec - It's a string formatted like <provider-type>:<data>

provider-type - KEY or KEOSD

Usare la chiave

```
signature-provider =  
PUBLIC_SIGNING_KEY=KEY:PRIVA'
```

Esempio

Firma del provider

=

EOS6MRyAjQq8ud7hVNYcfnVPJqcVps

Usare keosd

Possiamo usare keosd invece che la hard-defining key:

signature-provider = KEOSD:<data>

esempio

EOS6MRyAjQq8ud7hVNYcfnVPJqcVps

Definire una lista peer

```
# Default p2p port is 9876  
p2p-peer-address = 123.255.78.9:9876
```

Caricare i plugin richiesti

```
plugin = eosio::chain_plugin  
plugin = eosio::producer_plugin
```

Non-producing node

In questo caso dobbiamo avere installato nodeos da qualche parte.

Un non-producing node è un nodo che non è stato configurato per produrre blocchi, un caso d'uso può essere di fornire la sincronizzazione ad un http-

RPC API pubblica per gli sviluppatori o un endpoint dedicato privato per la propria dApp, tutto il procedimento è abbastanza semplice.

Settare i peer

Possiamo o settarli direttamente dal file config.ini

```
p2p-peer-address = 106.10.42.238:9876
```

oppure tramite linea di comando

```
--p2p-peer-  
address=106.10.42.238:9876
```

Ambiente di sviluppo

Ci sono diversi modi per configurare il proprio ambiente di sviluppo con nodeos, ed ovviamente la scelta dipende da come e quanto si vuole espandere il progetto, di seguito si elencano alcune soluzioni che possono essere adottate:

- local single-node testnet, questa è l'opzione per gli sviluppatori di smart contract, per chi vuole aspirare a diventare BP o operare come non-producing node, ha una configurazione semplice e poche richieste;

- local multi-node testnet, anche se

questa opzione è più tecnica per lo sviluppo di smart contract, questa potrebbe essere troppo per alcuni, infatti, questa è pensata soprattutto gli aspetti dello sviluppo del core come i benchmark, l'ottimizzazione e le sperimentazioni ed ovviamente per i curiosi.

Local single-node testnet

Dopo che avremo costruito il nostro progetto, il codice binario di nodeos dovrebbe essere presente nella cartella `build/programs/nodeos`, dove nodeos può essere avviato direttamente dalla cartella `build` usando

programs/nodeos/nodeos, oppure cambiare la cartella con cdprograms/nodeos e avviare nodeos da lì, in questa procedura avvieremo il comando dalla cartella programs/nodeos.

Con questo comando si può avviare il proprio single-node blockchain:

```
cd build/programs/nodeos
./nodeos -e -p eosio --plugin
eosio::chain_api_plugin --plugin
eosio::history_api_plugin
```

Quando si avvia nodeos di dovrebbe vedere un messaggio di log, simile a quanto riportato in basso, e questo significa che abbiamo prodotto con successo un blocco:

1575001ms thread-0
chain_controller.cpp:235
_push_block] initm #1 @2017-09-04T04:26:15 | 0 trx, 0 pending,
exectime_ms=0

1575001ms thread-0
producer_plugin.cpp:207
block_production_loop] initm generated
block #1 @ 2017-09-04T04:26:15 with
0 trxs 0 pending

1578001ms thread-0
chain_controller.cpp:235
_push_block] initc #2 @2017-09-04T04:26:18 | 0 trx, 0 pending,
exectime_ms=0

1578001ms thread-0
producer_plugin.cpp:207
block_production_loop] initc generated

block #2 @ 2017-09-04T04:26:18 with
0 trxs 0 pending

...

eosio generated block 046b9984...
#101527 @ 2018-04-01T14:24:58.000
with 0 trxs

eosio generated block 5e527ee2...
#101528 @ 2018-04-01T14:24:58.500
with 0 trxs

A questo punto nodeos sta lavorando su
un singolo producer, eosio.

Gli utenti più avanzati avranno
necessità di modificare la
configurazione e dove nodeos permette
di configurare la relativa cartella, che si
trova in questo percorso:

-Mac OS: ~/Library/Application\

Support/eosio/nodeos/config

-

Linux: `~/.local/share/eosio/nodeos/config`,

In quella cartella troveremo il file `genesis.json`, mentre se utilizziamo la linea di comando tramite `--config-dir` allora dovremmo copiare il relativo file `genesis.json` in quella cartella.

Nodeos ha bisogno che il file `config.ini` sia configurato correttamente per effettuare operazioni di significato, quindi all'avvio `nodeos` guarda nella cartella per vedere se è presente il file `config.ini` e se non trova nessun file allora ne crea uno di default dove poi potremmo aggiungere questi parametri

per farlo funzionare al meglio:

Enable production on a stale chain,
since a single-node test chain is pretty
much always stale

```
enable-stale-production = true
```

Enable block production with the
testnet producers

```
producer-name = eosio
```

Load the block producer plugin, so
you can produce blocks

```
plugin = eosio::producer_plugin
```

As well as API and HTTP plugins

```
plugin = eosio::chain_api_plugin
```

```
plugin = eosio::http_plugin
```

This will be used by the validation
step below, to view history

```
plugin = eosio::history_api_plugin
```

Adesso dovrebbe essere possibile avviare nodeos e vedere la produzione dei blocchi

```
./programs/nodeos/nodeos
```

Nodeos inoltre salva alcuni dati in una cartella data, dove la possiamo richiamare tramite il comando `--data-dir`, e dove si trova nei seguenti percorsi:

- Mac OS: `~/Library/Application Support/eosio/nodeos/data`

-

- Linux: `~/.local/share/eosio/nodeos/data`

Local multi-node testnet

Adesso andremo a settare 2 nodi sulla nostra macchina in locale così da metterli in comunicazione tra loro, ovviamente keosd, cleos e nodeos devono essere installati o come avviarli.

Avviamo la gestione del wallet

Nella prima finestra dobbiamo avviare keosd l'applicazione per la gestione del wallet:

```
keosd --http-server-address
```

127.0.0.1:8899

Una volta avviato, keosd mostrerà alcune informazioni iniziando con:

...

```
2493323ms thread-0
wallet_plugin.cpp:39
plugin_initialize ] initializing wallet
plugin
2493323ms thread-0
http_plugin.cpp:141
plugin_initialize ] host: 127.0.0.1 port:
8899
2493323ms thread-0
http_plugin.cpp:144
```

```
plugin_initialize    ] configured http to  
listen on 127.0.0.1:8899  
2493323ms          thread-0  
http_plugin.cpp: 213  
plugin_startup      ] start listening for  
http requests  
2493324ms          thread-0  
wallet_api_plugin.cpp: 70  
plugin_startup      ] starting  
wallet_api_plugin  
...
```

Dove alla porta 127.0.0.1:8899 il wallet è in ascolto ed indica che keosd è stato avviato correttamente e sulla porta corretta, se invece c'è qualche problema prima di avviare il tutto "starting wallet_api_plugin" allora dovremmo fare la diagnosi del problema e vedere

l'errore e poi riavviare. Una volta che keosd è avviato correttamente, lasciamo aperta la schermata ed apriamo una nuova finestra del terminale.

Creare il wallet di default

Nella nuova finestra usiamo cleos, l'utilità di linea di comando, per creare un wallet di default:

```
cleos --wallet-url http://127.0.0.1:8899  
wallet create --to-console
```

Cleos indicherà che è stato creato il wallet di default e ci fornirà la password per i futuri accessi, che

dovremmo salvarcela:

Creating wallet: default

Save password to use in the future to unlock this wallet.

Without password imported keys will not be retrievable.

"PW5JsmfYz2wrdUEotTzBamUCAunAA"

Caricare la chiave di eosio

Il lancio della blockchain privata è stata creata con le chiavi iniziali di default che devono essere caricate all'interno del wallet:

```
$ cleos --wallet-url  
http://127.0.0.1:8899 wallet import --
```

private-key

```
5KQwrPbwdL6PhXujxW37FSSQZ1Jiws  
imported private key for:  
EOS6MRyAjQq8ud7hVNYcfnVPJqcVps
```

Avviare il primo producer node

Adesso possiamo lanciare il primo producer node e nella terza finestra dobbiamo avviare:

```
nodeos --enable-stale-production --  
producer-name eosio --plugin  
eosio::chain_api_plugin --plugin  
eosio::net_api_plugin
```

Questo permette di creare uno

producer speciale, conosciuto come "bios" producer, e se tutto è andato per il verso giusto potremmo vedere l'output su nodeos che processa la creazione dei nuovi blocchi.

Avviare il secondo producer node

Ribadiamo che è necessario avviare il tutto dalla directory `EOSIO_SOURCE` e da dove avvieremo `./eosio_build.sh` per costruire il tutto. Per avviare un nuovo aggiuntivo bisogna prima caricare il contratto `eosio.bios`, dove questo contratto permette di avere il diretto controllo sulle risorse da allocare sugli account

ed accedere con permessi privilegiati sulle API richiamate; quindi ci dobbiamo recare sulla nostra seconda finestra precedente e carichiamo il seguente contratto:

```
cleos --wallet-url http://127.0.0.1:8899  
set contract eosio  
build/contracts/eosio.bios
```

Creeremo un account che diventerà a producer, usando il nome dell'account inita, dove per creare l'account dobbiamo generare le chiavi associate all'account che poi le dovremmo importare nel nostro wallet:

```
cleos create key
```

che ci genererà le seguenti chiavi

```
Private key:
5JgbL2ZnoEAhTudReWH1RnMuQS6DE
Public key:
EOS6hMjoWRF2L8x9YpeqtUEcsDKAy.
```

adesso dobbiamo importarle nel nostro wallet e se il tutto avrà successo avremmo verrà riportata la chiave pubblica dello stesso:

```
cleos --wallet-url http://127.0.0.1:8899
wallet import
5JgbL2ZnoEAhTudReWH1RnMuQS6DE
imported private key for:
EOS6hMjoWRF2L8x9YpeqtUEcsDKAy.
```

Adesso dobbiamo creare l'account inita che lo useremo come producer, e per creare un account ci servono 2 chiavi pubbliche, una per la chiave proprietaria e l'altra è la clave attiva:

```
cleos --wallet-url http://127.0.0.1:8899
create account eosio inita
EOS6hMjoWRF2L8x9YpeqtUEcsDKAy.
EOS6hMjoWRF2L8x9YpeqtUEcsDKAy.
executed transaction:
d1ea511977803d2d88f46deb554f5b6cce
352 bytes 102400 cycles
# eosio <=
eosio::newaccount
```

```
{"creator":"eosio","name":"inita","owner":  
{"threshold":1,"keys":  
[{"key":"EOS6hMjowWRF2L8x9YpeqtUE
```

In questo caso abbiamo creato un account che è stato concepito per la produzione dei blocchi.

Nella quarta finestra dobbiamo avviare una seconda istanza di nodeos, da notare che questa linea di comando è più lunga di quella usata precedentemente, proprio per evitare che il tutto collida con il primo nodeos:

```
nodeos --producer-name inita --plugin  
eosio::chain_api_plugin --plugin  
eosio::net_api_plugin --http-server-  
address 127.0.0.1:8889 --p2p-listen-
```

```
endpoint 127.0.0.1:9877 --p2p-peer-  
address 127.0.0.1:9876 --config-dir  
node2 --data-dir node2 --private-key  
["EOS6hMjoWRF2L8x9YpeqtUEcsDKL
```

L'output di questo nuovo nodo mostrerà una piccola attività fino alla registrazione dell'account inita come producer account, ma il vostro potrebbe essere leggermente differente:

```
2393147ms thread-0  
producer_plugin.cpp:176  
plugin_startup ] producer plugin:  
plugin_startup() end  
2393157ms thread-0  
net_plugin.cpp:1271
```



```
start_sync ] Catching up with
chain, our last req is 0, theirs is 8249
peer dhcp15.ocweb.com:9876 -
295f5fd
2393158ms thread-0
chain_controller.cpp:1402
validate_block_heade ]
head_block_time 2018-03-
01T12:00:00.000, next_block 2018-04-
05T22:31:08.500, block_interval 500
2393158ms thread-0
chain_controller.cpp:1404
validate_block_heade ] Did not produce
block within block_interval 500ms, took
3061868500ms)
2393512ms thread-0
producer_plugin.cpp:241
block_production_loo ] Not producing
```

block because production is disabled
until we receive a recent block (see: --
enable-stale-production)

2395680ms thread-0

net_plugin.cpp:1385

recv_notice] sync_manager got
last irreversible block notice

2395680ms thread-0

net_plugin.cpp:1271

start_sync] Catching up with
chain, our last req is 8248, theirs is
8255 peer dhcp15.ocicweb.com:9876 -
295f5fd

2396002ms thread-0

producer_plugin.cpp:226

block_production_loop] Previous result
occurred 5 times

2396002ms thread-0

```
producer_plugin.cpp:244  
block_production_loop ] Not producing  
block because it isn't my turn, its eosio
```

A questo punto il secondo nodeos è un producer idle, e per trasformarlo in un active producer, inita deve essere registrato come producer con il bios node, dove quest'ultimo performerà le azioni per aggiornare la tabella del producer:

```
cleos --wallet-url http://127.0.0.1:8899  
push action eosio setprods "{  
"schedule": [ {"producer_name":  
"inita"}, {"block_signing_key":  
"EOS6hMjoWRF2L8x9YpeqtUEcsDKA  
-p eosio@active  
executed transaction:
```

```
2cff4d96814752aefaf9908a7650e867dat  
272 bytes 105472 cycles  
# eosio <= eosio::setprods  
{ "version": 1, "producers":  
[ { "producer_name": "inita", "block_signir
```

Se tutto è andato per il meglio avrete configurato 2 nodi per la testnet, adesso potete vedere come il nodo originale non produce più I blocchi ma li riceve solamente ed è possibile verificare la cosa tramite il comando get info su ogni nodo:

primo nodo:

```
cleos get info
```

con questo output:

```
{
  "server_version": "223565e8",
  "head_block_num": 11412,
  "last_irreversible_block_num": 11411,
  "head_block_id":
"00002c94daf7dff456cd940bd585c4d9b",
  "head_block_time": "2018-04-
06T00:06:14",
  "head_block_producer": "inita"
}
```

secondo nodo:

```
cleos --url http://127.0.0.1:8889 get info
```

con questo output:

```
{
  "server_version": "223565e8",
  "head_block_num": 11438,
  "last_irreversible_block_num": 11437,
  "head_block_id":
"00002cae32697444fa9a2964e4db85b5e",
  "head_block_time": "2018-04-
06T00:06:27",
  "head_block_producer": "inita"
}
```

Riprodurre noeos

Nodeos fornisce numerose opzioni per riprodurre i blocchi della blockchain, questo può essere utile, per esempio, se

un nodo ha scaricato il file blocks.log da internet ed il nodo vuole usarlo per mettersi in pari con la rete o se vuole un punto specifico nella storia della blockchain.

Per riprodurre i dati si possono usare 2 metodi:

- da un file blocks.log, che contiene tutte le transazioni irreversibili sulla blockchain, dove tutte le istanze di nodeos sono scritte in quel file che si trova nella directory data/blocks relativa alla directory di nodeos, dove usando quel file è possibile avviare nodeos ricreando intera storia della blockchain localmente, senza caricare la rete;

- da un file snapshot, dove questi

vengono creati da un'istanza di nodeos operante, e contiene lo stato attuale della testa del blocco, e devono essere usati se quello che si vuole rappresentare è nella testa del blocco in maniera irreversibile, con questo metodo si inizia nodeos con un punto esatto di un blocco ma non si avrà tutta la storia risalente fino a quel blocco.

Prendere un blocks log

Il percorso di default del file blocks.log si trova nella directory data/blocks ma comunque possiamo specificare la directory con il comando -d [--data-dir], ma è possibile anche scaricare il file

in un altro modo anche tramite la semplice ricerca in rete.

Riprodurre da un blocks log

Una volta che abbiamo ottenuto il file, lo dobbiamo copiare nella relativa directory e fare un backup dello stesso per un utilizzo futuro e quindi avremmo una situazione del genere:

posizione	Nome
Data/blocks	Blocks.index
Data/blockes	Blocks.log

Data/blocks/reversible	Forkdb.dat
Data/blocks/reversible	Shared_memory
Data/blocks/reversible	Shared_memory

Possiamo anche usare `blocks-dir = "blocks"` nel file di configurazione oppure usare il comando `--blocks-dir`, per specificare dove trovare il file `blocks.log` che vogliamo riprodurre:

```
nodeos --replay-blockchain
-e -p eosio
--plugin eosio::producer_plugin
--plugin eosio::chain_api_plugin
--plugin eosio::http_plugin >>
nodeos.log 2>&1 &
```

Fare uno snapshot

Possiamo forzare un nodo nodeos a creare uno snapshot usando `create_snapshot` richiamando le RPC API che sono supportate all'interno del `producer_api_plugin`. Questo creerà il file `snapshot` nella directory `data/snapshot`, dove questo file ha il nome `<l'id della testa del blocco in esadecimale>.bin`, ed anche in questo caso si può anche scaricare il file dello snapshot cercandolo in rete.

Riprodurre da uno snapshot

Una volta ottenuto la copia dello snapshot la dobbiamo copiare nella relativa directory data/snapshots e fare un backup dello stesso per un utilizzo futuro e quindi avremmo una situazione del genere:

posizione	Nome	Azi
Data/snapshots	<l'id della testa del blocco in esadecimale>.bin	Sost il con quel nuov
Data/	*	elin

Possiamo usare snapshots-dir = "snapshots" nel file di configurazione

oppure usare il comando `--snapshots-dir`, per specificare dove trovare il file snapshot, usando il comando `--snapshot` e specificare il nome del file:

```
nodeos --snapshot yoursnapshot.name  
-e -p eosio  
--plugin eosio::producer_plugin  
--plugin eosio::chain_api_plugin  
--plugin eosio::http_plugin >>  
nodeos.log 2>&1 &
```

Quando riproduciamo da un file snapshot è raccomandato che tutti i dati siano rimossi, ma comunque se c'è il `blocks.log` questo deve contenere le informazioni dei blocchi che arrivino allo snapshot almeno, e se non ci sono le

relative informazioni lo snapshot creerà delle eccezioni e dove qualsiasi blocco reversibile disponibile sarà possibile:

blocks.log	Snapshot	Azione
Nessun blocks.log	Per il blocco irreversibile 2000	ok
Contiene i blocchi da 1 a 1999	Per il blocco irreversibile 2000	eccezione
Contiene i blocchi da 1 a 2001	Per il blocco irreversibile 2000	Ok - sarà ricreato dallo snapshot e fatto partire dal blocco

Quando si opera con il file snapshot, non si può passare argomenti del `--genesis-json` oppure `--genesis-timestamp` in `nodeos` dato che quelle informazioni sono caricate dal file snapshot.

Se il file `blocks.log` esiste, le informazioni del `genesis` che contiene saranno validate contro quelle dello snapshot e la riproduzione fallirà con un errore che i dati del `genesis` sono inconsistenti (per esempio i file sono di 2 blockchain diverse).

Alcune problematiche

database dirty flag set (likely due to

unclean shutdown): replay required

Nodeos ha bisogno di essere spento in modo corretto e per assicurare che questo avvenga, possiamo mandare una richiesta a SIGTERM, SIGQUIT or SIGINT o aspettare il processo di spegnimento, dove fallire questa procedura porta a quel tipo di errore e se capita bisogna avviare nodeos con --replay-blockahin.

memory does not match data error at restart

Se vediamo un errore del genere St9exception: content of memory does

not match data expected by executable
quando cerchiamo di avviare nodeos, è
consigliabile riavviare nodeos con una
delle seguenti opzioni (si può usare
nodeos --help per avere una lista
completa degli stessi):

Command Line Options for
eosio::chain_plugin:

--fix-reversible-blocks

recovers reversible block database if
that database
is in a bad state

--force-all-checks do not
skip any checks that can be
skipped while
replaying irreversible
blocks

--replay-blockchain clear
chain state database and replay
all blocks

--hard-replay-blockchain
clear chain state database, recover as
many blocks
as possible from the block
log, and then
replay those blocks

--delete-all-blocks clear
chain state database and block
log

**could not grow database file to
requested size. error**

Avviare nodeos con --shared-memory-

size-mb 1024 dove 1GB di memoria condivisa permette circa mezzo milione di transazioni.

Che versione di EOSIO sto utilizzando?

Se è utilizzato di default il comando `cleos get info` darà un output che contiene un campo chiamato `server_version`. Se il nostro `nodeos` non usa parametri di default, allora dobbiamo conoscere l'url di `nodeos` e si usa il seguente comando:

```
cleos --url http://localhost:8888 get info
```

e dobbiamo focalizzarci solo sulla versione:

```
cleos --url http://localhost:8888 get info  
| grep server_version
```

Modalità lettura

EOSIO fornisce una serie di servizi ed interfacce che permette ai contratti sviluppati di persistere tra le varie azioni e transazioni, dove i contratti possono usare questi servizi ed interfacce per scopi differenti, come ad esempio controllare il saldo degli utenti di un determinato database.

Ogni istanza di nodeos tiene in memoria il database, così i contratti possono leggere e scrivere i dati e fornisce anche l'accesso a questi dati sull'http RPC API per la lettura del database.

In qualsiasi momento è possibile richiedere quei dati:

- speculative, questo include gli effetti delle transazioni confermate e non;
- head, questo include solamente l'effetto delle transazioni confermate, dove le transazioni non confermate vengono processate ma non sono incluse;
- reand-only, questo include solo l'effetto delle transazioni confermate.

Una transazione è considerata confermata quando un'istanza di nodeos è stata ricevuta, processata e scritta dentro un blocco nella blockchain, per esempio nella testa del blocco o in blocchi recenti.

Speculative

I client come cleos e le RPC API, vedono il database come l'attuale testa del blocco più i cambiamenti fatti da tutte le transazioni note a quel nodo ma che potenzialmente non sono incluse nella catena, come quelle non confermate per esempio.

Lo speculative mode ha una bassa latenza ed è molto fragile, e quindi non c'è nessuna garanzia che le transazioni riflettano in cui saranno incluse le transazioni nella catena o l'ordine stesso. In questa modalità nodeos è in grado di eseguire le transazioni TaPoS puntandole a qualsiasi blocco valido in un fork considerandolo il miglior fork per questo nodo.

Head

Dato che la testa del blocco corrente non è ancora irreversibile ed un piccolo fork è possibile, la lettura in questa modalità potrebbe essere inaccurata se nodeos cambia per un fork migliore, cosa che accade anche nella modalità speculative.

In questa modalità nodeos è in grado di eseguire le transazioni TaPoS puntandole a qualsiasi blocco valido in un fork considerandolo il miglior fork per questo nodo.

Read mode

Questa modalità è specifica per usare il comando `--read-mode` con le informazioni di `eosio::chain_plugin`.

Read-only

Questa modalità non include i

cambiamenti fatti dalle transazioni conosciute al nodo ma non incluse nella catena, come quelle non confermate.

Irreversible

Quando nodeos è configurato in modalità lettura irreversibile, potrà tracciare gli ultimi blocchi recenti nel database, ma lo stato sarà indietro alla testa del blocco, e dove viene chiamato in questi casi fork DB head per identificare lo stato dell'ultimo blocco irreversibile.

Piccola nota

La piattaforma EOSIO salva le informazioni della blockchain in diverse strutture di dati ed alcune di queste sono descritte di seguito, ed in queste descrizioni i producing node sono istanze di nodeos che sono fatte girare dai BP che sono coloro che producono i blocchi:

- block log, contiene il log dei blocchi che sono scritte sul disco e contiene tutti i blocchi irreversibili;
- reversible_blocks, contiene i blocchi che sono stati scritti nella blockchain ma che non sono ancora diventati irreversibili;

- chain state o database, salva lo stato della blockchain per ogni blocco (informazioni dell'account, transazioni e così via) e dopo che il blocco diventa irreversibile non lo vediamo più in questo stato;

- pending block, contiene le transazioni come sono state processate nel blocco, dove potrebbe essere o diventare la testa del blocco, dove se questa istanza di nodeos è il producing node allora il pending block viene distribuito alle altre istanze di nodeos;

- la testa del blocco è l'ultimo blocco scritto sulla blockchain, salvato in `reversible_blocks`.

Logging

Il logging di nodeos è controllato dal file logging.json, che di solito si trova in --config-dir, stessa directory del file config.ini, e questo percorso può essere definito usando -l oppure --logconf quando si avvia nodeos.

```
./nodeos --help
```

...

Application Command Line Options:

...

--config-dir arg Directory containing configuration files such as config.ini

-l [--logconf] arg (=logging.json) Logging configuration file name/path for

library users

Il file logging.json può essere definito per appender e loggers.

Appenders

La libreria logging costruita in EOSIO supporta alcuni tipi di appender:

- console

Questo mostrerà l'output del messaggio di log e le opzioni per la configurazione sono:

name - arbitrary name to identify instance for use in loggers

type - "console"

stream - "std_out" or "std_err"

level_colors - maps a log level to a colour.

-- level - see logging levels below.

-- color - may be one of ("red", "green", "brown", "blue", "magenta", "cyan", "white", "console_default")

enabled - bool value to enable/disable the appender.

esempio:

```
{  
  "name": "consoleout",  
  "type": "console",  
  "args": {  
    "stream": "std_out",
```

```
"level_colors": [{
  "level": "debug",
  "color": "green"
}, {
  "level": "warn",
  "color": "brown"
}, {
  "level": "error",
  "color": "red"
}
]
},
"enabled": true
}
```

- gelf

Questo manda un messaggio di log a Graylog, piattaforma per la collezione, indicizzazione ed analisi dei messaggi I log, e le opzioni per la configurazione sono:

name - nome arbitrario per identificare l'istanza da usare nel logger

type - "gelf"

endpoint - indirizzo ip e numero di porta

host - Graylog hostname, coem ci identifichiamo al Graylog.

enabled - valore booleano per abilitare/disabilitare l'appender.

esempio:


```
{
  "name": "net",
  "type": "gelf",
  "args": {
    "endpoint":
"104.198.210.18:12202",
    "host": <YOURNAMEHERE IN
QUOTES>
  },
  "enabled": true
}
```

Loggers

La libreria logging costruita in EOSIO supporta alcuni tipi di logger:

default, il logger di default, sempre attivato.

net_plugin_impl, logging dettagliato per I plugin della net.

bnet_plugin, logging dettagliato per I plugin della bnet.

producer_plugin, logging dettagliato per I plugin del producer.

transaction_tracing, logging dettagliato che emette verdetti da un nodo affidabile della rete p2p.

e le opzioni per la configurazione sono:

name - deve corrispondere con uno dei nomi descritti in precedenza.

level - guarda il livello in basso del logging.

enabled - valore booleano per abilitare/disabilitare il logger.

additivity - true or false

appenders

-- list of appenders by name (name in the appender configuration)

esempio:

```
{  
  "name": "net_plugin_impl",  
  "level": "debug",  
  "enabled": true,  
  "additivity": false,  
  "appenders": [  
    "net"  
  ]  
}
```

net_plugin_impl, bnet_plugin, producer_] non sono abilitati a meno che esplicitamente abilitati nel file logging.json.

Logging levels

Ci sono diversi livelli di logging:

all

debug

info

warn

error

off

un esempio di file logging.json:

```
{
```

```
"includes": [],  
"appenders": [{  
  "name": "consoleout",  
  "type": "console",  
  "args": {  
    "stream": "std_out",  
    "level_colors": [{  
      "level": "debug",  
      "color": "green"  
    }], {  
      "level": "warn",  
      "color": "brown"  
    }], {  
      "level": "error",  
      "color": "red"  
    }  
  }  
}
```

```
}
```

```
]
```

```
},
```

```
"enabled": true
```

```
},{
```

```
"name": "net",
```

```
"type": "gelf",
```

```
"args": {
```

```
"endpoint": "10.10.10.10",
```

```
"host": "test"
```

```
},
```

```
"enabled": true
```

```
}
```

```
],
```

```
"loggers": [{
```

```
"name": "default",  
"level": "info",  
"enabled": true,  
"additivity": false,  
"appenders": [  
  "consoleout",  
  "net"  
]  
}, {  
  "name": "net_plugin_impl",  
  "level": "debug",  
  "enabled": true,  
  "additivity": false,  
  "appenders": [  
    "net"
```

]

}

]

}

Guida cleos

Cleos è uno strumento a linea di comando che si interfaccia con le REST API esposte da nodeos, e per usare cleos bisogna avere un end point (indirizzo IP e un numero di porta) ad un'istanza di nodeos ed anche configurare nodeos per caricare eosio::chain_api_plugin, ovviamente cleos contiene tutta la documentazione per i suoi comandi, dove per visualizzarli basta avviarlo senza argomenti:

Command Line Interface to EOSIO Client

Usage: `./programs/cleos/cleos`
[OPTIONS] SUBCOMMAND

Options:

`-h,--help` Print this help message and exit

`-u,--url TEXT=http://localhost:8888/`
the http/https URL where nodeos is running

`--wallet-url TEXT=http://localhost:8900/`
the http/https URL where keosd is running

`-r,--header` pass specific HTTP header; repeat this option to pass multiple headers

-n,--no-verify don't verify peer
certificate when using HTTPS
-v,--verbose output verbose
actions on error

Subcommands:

version Retrieve version
information

create Create various
items, on and off the blockchain

get Retrieve various
items and information from the
blockchain

set Set or update
blockchain state

transfer Transfer EOS from
account to account

net	Interact with local
p2p network connections	
wallet	Interact with local
wallet	
sign	Sign a transaction
push	Push arbitrary
transactions to the blockchain	
multisig	Multisig contract
commands	
system	Send eosio.system
contract action to the blockchain.	

Per avere aiuto con un particolare sottocomando, bisogna avviarlo senza argomenti:

Create various items, on and off the blockchain

Usage: ./cleos create SUBCOMMAND

Subcommands:

key Create a new
keypair and print the public and private
keys

account Create an account,
buy ram, stake for bandwidth for the
account

Create an account, buy ram, stake for
bandwidth for the account

Usage:

/programs/cleos/cleos create account

Options:

creator name OwnerKey [ActiveKey]

Positionals:

creator TEXT The name of
the account creating the new account
(required)

name TEXT The name of
the new account (required)

OwnerKey TEXT The owner
public key for the new account
(required)

ActiveKey TEXT The active
public key for the new account

Options:

- h,--help Print this help message and exit
- x,--expiration set the time in seconds before a transaction expires, defaults to 30s
- f,--force-unique force the transaction to be unique. this will consume extra bandwidth and remove any protections against accidentally issuing the same transaction multiple times
- s,--skip-sign Specify if unlocked wallet keys should be used to sign transaction
- j,--json print result as json
- d,--dont-broadcast don't

broadcast transaction to the network
(just print to stdout)

-r,--ref-block TEXT set the
reference block num or block id used for
TAPOS (Transaction as Proof-of-Stake)

-p,--permission TEXT ... An account
and permission level to authorize, as in
'account@permission'

--max-cpu-usage-ms UINT set an
upper limit on the milliseconds of cpu
usage budget, for the execution of the
transaction (defaults to 0 which means
no limit)

--max-net-usage UINT set an upper
limit on the net usage budget, in bytes,
for the transaction (defaults to 0 which
means no limit)

Connettersi ad un host/porta non di default

Prima di seguire la procedura è necessario che eosiocpp, cleos, nodeos e keosd siano aggiunti al nostro \$PATH, di seguito una breve guida sullo stesso:

Aliasing EOSIO components

Per il docker

```
alias cleos='docker exec -it eosio  
/opt/eosio/bin/cleos -u  
http://localhost:8888'
```

```
alias keosd='docker exec -it eosio  
/opt/eosio/bin/keosd --wallet-url  
http://localhost:8888'
```

Determinare il nostro percorso

- cercare il nostro percorso ad eos

```
$ cd eos  
$ pwd  
/path/to/eos
```

Usare alias

Aprire il file `~/.bash_profile` con un text

editor e rimpiazzare YOURPATH con quello nostro

```
#NODEOS
```

```
alias
```

```
nodeos=YOURPATH/build/programs/no
```

```
#CLEOS
```

```
alias
```

```
cleos=YOURPATH/build/programs/cleo
```

```
#KEOSD
```

```
alias
```

```
keosd=YOURPATH/build/programs/keo
```

```
#EOSIOCPP
```

```
alias
```

```
eosiocpp=YOURPATH/build/tools/eosio
```

Aggiungere a PATH

#VIM

vi ~/.bash_profile

#NANO

nano ~/.bash_profile

#PICO

pico ~/.bash_profile

#ATOM

atom ~/.bash_profile

#Default text editor (mac)

open ~/.bash_profile

#CLEOS

export

PATH=YOURPATH/build/programs/cle

#NODEOS

export

PATH=YOURPATH/build/programs/noc

```
#KEOSD
```

```
export
```

```
PATH=YOURPATH/build/programs/keosd
```

```
#EOSIOCPP
```

```
export
```

```
PATH=YOURPATH/build/tools/eosiocpp
```

Adesso avremo accesso a cleos, nodeos, keosd, eosiocpp da qualsiasi parte nel sistema:

```
$ cleos
```

```
ERROR: RequiredError: Subcommand required
```

```
Command Line Interface to EOSIO Client
```

```
Usage:
```

cleos

Options:

- h,--help Print this help message and exit
- H,--host TEXT=localhost the host where nodeos is running
- p,--port UINT=8888 the port where nodeos is running
- wallet-host TEXT=localhost the host where keosd is running
- wallet-port UINT=8888 the port where keosd is running
- v,--verbose output verbose actions on error

Subcommands:

version information	Retrieve version
create items, on and off the blockchain	Create various
get items and information from the blockchain	Retrieve various
set blockchain state	Set or update
transfer account to account	Transfer EOS from
net p2p network connections	Interact with local
wallet wallet	Interact with local
sign	Sign a transaction
push	Push arbitrary

transactions to the blockchain

multisig
commands

Multisig contract

Una volta configurato il tutto procediamo come segue.

Da notare che se nessun argomento opzionale è usato, per esempio --url e --wallet-url, cleos automaticamente cercherà di connettersi localmente al nodo di eos che sta funzionando, cioè nodeos.

Comandi:

- connettersi a nodeos:

```
cleos --url http://127.0.0.1:8888  
${subcommand}
```

- connettersi a keosd:

```
cleos --wallet-url  
http://test1.eos.io:8888 ${subcommand}
```

ricordiamo inoltre che `--wallet-url` e `--url` devono essere usati per ogni esecuzione di `cleos` per permettere ai comandi di operare con il nodo desiderato.

A questo punto `keosd` parte automaticamente ed è possibile, mentre si sta sviluppando e testando, che `keosd` si avvii manualmente (non da `cleos`) a ci si ritrovi ad avere più istanze di `keosd` e quando succede è possibile che `cleos`

non trovi il giusto set di chiavi; per vedere se ci sono più istanze di keosd e quali porte stanno usando possiamo vederle in questo modo:

```
$ pgrep keosd | xargs printf " -p %d" |  
xargs lsof -Pani
```

COMMAND	PID	USER	FD
TYPE		DEVICE	SIZE/OFF
NODE	NAME		
keosd	49590	tutorial	6u IPv4
		0x72cd8ccf8c2c2d03	0t0 TCP
		127.0.0.1:8900 (LISTEN)	
keosd	62812	tutorial	7u IPv4
		0x72cd8ccf90428783	0t0 TCP
		127.0.0.1:8899 (LISTEN)	

Aggiungere eosio.code ad un active authority con cleos helper

Quando sviluppiamo un contratto può essere necessario per lo stesso di avere l'abilità di trasmettere le azioni, e per fare questo è necessario che questo usi l'autorità active; tuttavia, per ragioni di sicurezza, i contratti non possono essere firmati con la loro active authority a meno che il contratto non è stato configurato in quel modo, eosio.code è una pseudo autorità che garantisce al

contratto l'active authority.

Aggiungere eosio.code ad un contratto active authority

```
cleos set account permission  
YOURCONTRACT active --add-code
```

Rimuovere eosio.code da un contratto active authority

```
cleos set account permission  
YOURCONTRACT active --remove-  
code
```

Quando utilizziamo quei comandi, --add-code e --remove-code, cleos ottiene

il permesso corrente dall'account per aggiungere o rimuovere YOURCONTRACT@eosio.code da un active permission.

Comandi di cleos

- version client

recupera le informazioni del client

Positionals:

none

Usage:

```
$ ./cleos version client
```

- pack_transaction

da un file json firmato lo trasforma in un pacchetto

Positionals:

transaction TEXT - il file formato json (stringa)

Options:

--pack-action-data - impacchetta tutte le azioni di dati dentro la transazione, deve interagire con nodeos

Usage:

```
cleos convert pack_transaction '{
  "expiration": "2018-08-02T20:24:36",
  "ref_block_num": 14207,
  "ref_block_prefix": 1438248607,
  "max_net_usage_words": 0,
  "max_cpu_usage_ms": 0,
  "delay_sec": 0,
  "context_free_actions": [],
  "actions": [{
    "account": "eosio",
    "name": "newaccount",
    "authorization": [{
```



```
    "actor": "eosio",
    "permission": "active"
  }
],
                                     "data":
"0000000000ea305500a6823403ea3055
  }
],
"transaction_extensions": []
}'
```

Output:

```
{
  "signatures": [],
  "compression": "none",
  "packed_context_free_data": ""
```

```
        "packed_trx":  
"8468635b7f379feeb9550000000001000  
}
```

- **unpack_transaction**

da un pacchetto ad un file firmato json

Positionals:

transaction TEXT - il pacchetto della transazione json (stringa contenente packed_trx e gli eventuali campi di opzioni)

Options:

--unpack.action-data - spacchetta tutte le azioni dei dati all'interno della transazione, è necessario interagire con nodeos

Usage:

```
cleos convert unpack_transaction '{
  "signatures": [
    "SIG_K1_KmRbWahefwxs6uyCGNR6w
  ],
  "compression": "none",
  "packed_context_free_data": "",
  "packed_trx":
```

```
"8468635b7f379feeb95500000000001000  
'
```

Output:

```
{  
  "expiration": "2018-08-02T20:24:36",  
  "ref_block_num": 14207,  
  "ref_block_prefix": 1438248607,  
  "max_net_usage_words": 0,  
  "max_cpu_usage_ms": 0,  
  "delay_sec": 0,  
  "context_free_actions": [],  
  "actions": [{  
    "account": "eosio",  
    "name": "newaccount",  
    "authorization": [{  
      "actor": "eosio",  
      "permission": "active"  
    }  
  ]  
}
```

```
    }  
  ],  
    "data":  
    "0000000000ea305500a6823403ea3055"  
    }  
  ],  
  "transaction_extensions": [],  
  "signatures": [  
    "SIG_K1_KmRbWahefwxs6uyCGNR6w"  
  ],  
  "context_free_data": []  
}
```

- pack_action_data

da un json action data ad una forma
impacchettata

Positionals:

account TEXT - il nome dell'account che hosta il contratto

name TEXT - il nome della funzione che è chiamata da questa azione

packed_action_data TEXT - i dati dell'azione espresse come json

Options:

none

Usage:

```
cleos convert pack_action_data eosio  
unlinkauth '{"account":"test1",  
"code":"test2", "type":"eosioeosio"}'
```

Output:

```
000000008090b1ca0000000000091b1ca0
```

- unpack_avtion_data

Positionals:

account TEXT - il nome dell'account che
hosta il contratto

name TEXT - il nome della funzione che
è chiamata da questa azione

unpacked_action_data TEXT - i dati
dell'azione espresse come json

Options:

none

Usage:

```
cleos convert unpack_action_data eosio  
unlinkauth  
000000008090b1ca000000000091b1ca0
```

Output:

```
{  
  "account": "test1",  
  "code": "test2",  
  "type": "eosioeosio"  
}
```


- get info

prende le informazioni della blockchain corrente.

Positionals:

questo comando non accetta altri parametri

Options:

none

Usage:

\$./cleos get info

}

- **get block**

recupera un blocco completo dalla blockchain.

Positional:

block TEXT - il numero o l'ID del blocco da recuperare

Options:

none

Usage:

\$./cleos get block 1

- get account

Recupera un account dalla blockchain.

Positionals:

name TEXT - il nome dell'account da recuperare

Options:

-j, --json - l'output in formato json

Usage:

prendere dati formattati per utilizzarli con eosio

\$ cleos get account eosio

privileged: true

permissions:

owner 1: 1

EOS6MRyAjQq8ud7hVNYcfnVPJqcVps

active 1: 1

EOS6MRyAjQq8ud7hVNYcfnVPJqcVps

memory:

quota: -1 bytes used: 1.22

Mb

net bandwidth: (averaged over 3 days)

used: -1 bytes

available: -1 bytes

limit: -1 bytes

cpu bandwidth: (averaged over 3 days)

used: -1 us
available: -1 us
limit: -1 us

producers: <not voted>

prendere dati json per utilizzarli con
eoso

```
$ cleos get account eosio --json
```

```
{  
  "account_name": "eosio",  
  "privileged": true,  
    "last_code_update": "2018-05-  
23T18:00:25.500",  
    "created": "2018-03-  
02T12:00:00.000",  
  "ram_quota": -1,  
  "net_weight": -1,  
  "cpu_weight": -1,
```



```
"net_limit": {
  "used": -1,
  "available": -1,
  "max": -1
},
"cpu_limit": {
  "used": -1,
  "available": -1,
  "max": -1
},
"ram_usage": 1279625,
"permissions": [{
  "perm_name": "active",
  "parent": "owner",
  "required_auth": {
    "threshold": 1,
    "keys": [{
```

```
"key":
```

```
"EOS6MRyAjQq8ud7hVNYcfnVPJqcVp
  "weight": 1
}
],
"accounts": [],
"waits": []
}
}, {
  "perm_name": "owner",
  "parent": "",
  "required_auth": {
    "threshold": 1,
    "keys": [ {
      "key":
"EOS6MRyAjQq8ud7hVNYcfnVPJqcVp
  "weight": 1
}
],
```

```
    "accounts": [],
    "waits": []
  }
},
"total_resources": null,
"delegated_bandwidth": null,
"voter_info": {
  "owner": "eosio",
  "proxy": "",
  "producers": [],
  "staked": 0,
  "last_vote_weight":
"0.00000000000000000000",
  "proxied_vote_weight":
"0.00000000000000000000",
  "is_proxy": 0,
  "deferred_trx_id": 0,
```

```
    "last_unstake_time": "1970-01-01T00:00:00",  
    "unstaking": "0.0000 SYS"  
  }  
}
```

- get code

Recupera il codice e l'ABI per un account.

Positionals:

name TEXT - il nome dell'account da dove deve essere preso il codice

Options:

-c, --code TEXT - il nome del file da

salvare il contratto .wast to

- a, --abi TEXT - il nome del file da
salvare il contratto . abi to

Usage:

```
$ cleos get code eosio.token
```

```
code hash:  
f675e7aeffb562c033acfaf33eadff255dac
```

```
$ cleos get code eosio.token -a  
eosio.token.abi
```

```
code hash:  
f675e7aeffb562c033acfaf33eadff255dac  
saving abi to eosio.token.abi
```

```
$ cleos get code eosio.token -c  
eosio.token.wast
```

code hash:
f675e7aeffb562c033acfaf33eadff255dac
saving wast to eosio.token.wast

- get table

Recupera il contenuto da una tabella database.

Positionals:

contract TEXT - il contratto che ha la tabella

scope TEXT - la ricerca all'interno del contratto dove la tabella si trova

table TEXT - il nome della tabella come specificato dal contratto abi

Options:

-b, --binary UINT - riporta il valore come binario piuttosto che usare l'abi per interpretare il json

-l, --limit UINT - il numero massimo di righe da riportare

-L, --lower TEXT - rappresentazione del json dei valori inferiori delle chiavi, di default dall'ultimo

--show-payer BOOLEAN - mostra chi ha pagato la RAM (default: false)

-r, --reverse BOOLEAN - interagisce in ordine inverso (default: false)

-u, --upper TEXT - rappresentazione del json dei valori superiori delle chiavi, di default dall'ultimo

--index TEXT - l'indice dei numeri, 1 - primario (primo), 2 - indice secondario (definito dal multi_index) e così via, dove il numero o il nome dell'indice può essere specificato per esempio secondario oppure 2

--key-type TEXT - il tpo di chiave dell'indice, solo supporto primario (i64) gli altri sono i128, i256, float64, float128, sha256, tipi speciali di nomi indica il nome dell'account

Usage:

Trova il nome all'asta con l'offerta più bassa.

```
cleos get table eosio eosio namebids --  
key-type i64 --index 2 -r -l 1
```



```
{
  "rows": [{
    "newname": "com",
    "high_bidder": "a123",
    "high_bid": 100000,
    "last_bid_time":
"1541667021500000"
  }
],
  "more": true
}
```

oppure mostra tutti i nomi dell'asta dal più grande al più piccolo, incluse le informazioni sulla RAM pagata

```
cleos get table eosio eosio namebids --
```

```
key-type i64 --index 2 -r --show-payer
```

```
{
```

```
  "rows": [{
```

```
    "data": {
```

```
      "newname": "com",
```

```
      "high_bidder": "a123",
```

```
      "high_bid": 100000,
```

```
      "last_bid_time":
```

```
"1541667021500000"
```

```
    },
```

```
    "payer": "a123"
```

```
  }, {
```

```
    "data": {
```

```
      "newname": "abc",
```

```
      "high_bidder": "a123",
```

```
      "high_bid": 110000,
```

```
      "last_bid_time":
```

```
"1541667021500000"
```

```
},
```

```
"payer": "a123"
```

```
},{
```

```
"data": {
```

```
  "newname": "ddd",
```

```
  "high_bidder": "a123",
```

```
  "high_bid": 120000,
```

```
  "last_bid_time":
```

```
"1541667021500000"
```

```
},
```

```
"payer": "a123"
```

```
},{
```

```
"data": {
```

```
  "newname": "zoo",
```

```
  "high_bidder": "a123",
```

```
  "high_bid": 9990000,
```

```
  "last_bid_time":
```

```
"1541667022000000"
```

```
    },  
    "payer": "a123"  
  }  
],  
"more": false  
}
```

- **get currency balance**

Recupera il saldo di un account per una determinata moneta.

Positionals:

contract TEXT - il contratto che opera la moneta

account TEXT - l'account per la

richiesta del saldo

symbol TEXT - il simbolo della moneta

se il contratto opera con più monete

Options:

none

Usage:

prende il bilancio di eosio dal contratto
eosio.token per il simbolo SYS

```
$ cleos get currency balance eosio.token  
eosio SYS
```

```
999999920.0000 SYS
```

- get currency stats

Recupera le statistiche per una data moneta.

Positionals:

contract TEXT - il contratto che opera la moneta

symbol TEXT - il simbolo della moneta
se il contratto opera con più monete

Options:

none

Usage:

recupera le statistiche per il token SYS dal contratto eosio.token

```
$ cleos get currency stats eosio.token  
SYS  
{  
  "SYS": {  
    "supply": "1000000000.0000 SYS",  
    "max_supply": "10000000000.0000  
SYS",  
    "issuer": "eosio"  
  }  
}
```

- get account

Recupera tutti gli account associati ad

una determinata chiave pubblica

Positionals:

public_key TEXT - la chiave pubblica per recuperare l'account

Options:

none

Usage:

```
$ cleos get accounts  
EOS8mUftjXepGzdQ2TaCduNuSPAfXJ.  
{  
  "account_names": [  
    "testaccount"
```



```
]
}
```

- get servants

Positionals:

account TEXT - si riferisce all'account da recuperare creato da quell'account

Options:

none

Usage:

\$ cleos get servants eosio

- get transactions

Recupera una transazione dalla blockchain.

Positionals:

id TEXT - id della transazione da recuperare

Options:

none

Usage:

```
$ ./cleos get transaction
```

Output:

esempio

```
$ ./cleos get transaction
```

```
eb4b94b72718a369af09eb2e7885b3f494
{
    "transaction_id":
"eb4b94b72718a369af09eb2e7885b3f49
"processed": {
    "refBlockNum": 2206,
    "refBlockPrefix": 221394282,
    "expiration": "2017-09-
05T08:03:58",
    "scope": [
        "inita",
        "tester"
    ],
    "signatures": [
"1f22e64240e1e479eee6ccbbd79a29f1a
    ],
    "messages": [ {
```

```
"code": "eos",  
"type": "transfer",  
"authorization": [{  
  "account": "inita",  
  "permission": "active"  
}
```

```
],  
"data": {  
  "from": "inita",  
  "to": "tester",  
  "amount": 1000,  
  "memo": ""  
},
```

```
                                "hex_data":  
"000000008040934b00000000c84267a1  
}
```

```
],  
"output": [{
```

```
"notify": [{
  "name": "tester",
  "output": {
    "notify": [],
    "sync_transactions": [],
    "async_transactions": []
  }
}], {
  "name": "inita",
  "output": {
    "notify": [],
    "sync_transactions": [],
    "async_transactions": []
  }
}],
"sync_transactions": [],
"async_transactions": []
```

```
}  
]  
}  
}
```

- get actions

Recupera tutte le azioni di uno specifico account, possiamo anche richiedere la lista delle azioni che hanno come account come ricevitori, i soli account richiamabili saranno quelli del ricevitore nell'opzione --filter-on receiver:action:actor

Positionals:

account_name TEXT - il nome dell'account che vogliamo interrogare

Options:

none

Usage:

queste azioni appartengono ad eosio ma le vostre saranno diverse:

```
$ ./cleos get actions eosio.token
```

```
#      seq      when
```

```
contract::action => receiver      trx id...
```


args

976 2018-06-01T19:54:05.000

eosio.token::transfer => eosio.token
1d1fe154...

{"from":"useraaaaaaaae","to":"useraaaaaa

977 2018-06-01T19:54:05.000

eosio.token::transfer => eosio.token
a0c9e5bc...

{"from":"useraaaaaaaaab","to":"useraaaaaa

978 2018-06-01T19:54:05.000

eosio.token::transfer => eosio.token
3749d0d1...

{"from":"useraaaaaaaaab","to":"useraaaaaa

979 2018-06-01T19:54:05.000

eosio.token::transfer => eosio.token
dda205b0...

{"from":"useraaaaaaaai","to":"useraaaaaa

```
# 980 2018-06-01T19:54:05.000
eosio.token::transfer => eosio.token
14089e9b...
{"from":"useraaaaaab","to":"useraaaaa
# 981 2018-06-01T19:54:05.000
eosio.token::transfer => eosio.token
6882cefc...
{"from":"useraaaaaaj","to":"useraaaaa
...
```

- get abi

Recupera l'ABI da un account.

Positionals:

name TEXT - il nome dell'account da

dove l'abi deve essere recuperato

Options:

-f, --file TEXT - il nome del file dove salvare il contratto .abi invece di scriverlo sulla console

esempio:

```
cleos get abi eosio
```

```
{  
  "version": "eosio::abi/1.1",  
  "types": [],  
  "structs": [{  
    "name": "abi_hash",  
    "base": "",  
    "fields": [{
```

```
    "name": "owner",
    "type": "name"
  }, {
    "name": "hash",
    "type": "checksum256"
  }
]
}, {
  "name": "authority",
  "base": "",
  "fields": [ {
    "name": "threshold",
    "type": "uint32"
  }, {
    "name": "keys",
    "type": "key_weight[]"
  }, {
    "name": "accounts",
```

"type":

```
"permission_level_weight[]"  
  }, {  
    "name": "waits",  
    "type": "wait_weight[]"  
  }  
]  
{  
  "name": "bid_refund",  
  "base": "",  
  "fields": [{  
    "name": "bidder",  
    "type": "name"  
  }, {  
    "name": "amount",  
    "type": "asset"  
  }  
]  
]
```

```
}, {  
  "name": "bidname",  
  "base": "",  
  "fields": [{  
    "name": "bidder",  
    "type": "name"  
  }], {  
    "name": "newname",  
    "type": "name"  
  }], {  
    "name": "bid",  
    "type": "asset"  
  }  
]  
}, {  
  "name": "bidrefund",  
  "base": "",  
  "fields": [{
```

```
    "name": "bidder",
    "type": "name"
  }, {
    "name": "newname",
    "type": "name"
  }
]
}, {
  "name": "block_header",
  "base": "",
  "fields": [ {
    "name": "timestamp",
    "type": "uint32"
  }, {
    "name": "producer",
    "type": "name"
  }, {
    "name": "confirmed",
```

```
    "type": "uint16"  
  }, {  
    "name": "previous",  
    "type": "checksum256"  
  }, {  
    "name": "transaction_mroot",  
    "type": "checksum256"  
  }, {  
    "name": "action_mroot",  
    "type": "checksum256"  
  }, {  
    "name": "schedule_version",  
    "type": "uint32"  
  }, {  
    "name": "new_producers",  
    "type": "producer_schedule?"  
  }  
]  
]
```



```
}, {  
  "name": "blockchain_parameters",  
  "base": "",  
  "fields": [{  
    "name": "max_block_net_usage",  
    "type": "uint64"  
  }], {  
    "name":  
"target_block_net_usage_pct",  
    "type": "uint32"  
  }], {  
    "name":  
"max_transaction_net_usage",  
    "type": "uint32"  
  }], {  
    "name":  
"base_per_transaction_net_usage",  
    "type": "uint32"
```

```
}, {  
  "name": "net_usage_leeway",  
  "type": "uint32"  
}, {
```

```
  "name":  
"context_free_discount_net_usage_num",  
  "type": "uint32"  
}, {
```

```
  "name":  
"context_free_discount_net_usage_den",  
  "type": "uint32"  
}, {
```

```
  "name": "max_block_cpu_usage",  
  "type": "uint32"  
}, {
```

```
  "name":  
"target_block_cpu_usage_pct",  
  "type": "uint32"
```

}, {

"name":

"max_transaction_cpu_usage",

"type": "uint32"

}, {

"name":

"min_transaction_cpu_usage",

"type": "uint32"

}, {

"name":

"max_transaction_lifetime",

"type": "uint32"

}, {

"name":

"deferred_trx_expiration_window",

"type": "uint32"

}, {

"name": "max_transaction_delay",

```
"type": "uint32"
```

```
}, {
```

```
"name":
```

```
"max_inline_action_size",
```

```
"type": "uint32"
```

```
}, {
```

```
"name":
```

```
"max_inline_action_depth",
```

```
"type": "uint16"
```

```
}, {
```

```
"name": "max_authority_depth",
```

```
"type": "uint16"
```

```
}
```

```
]
```

```
}, {
```

```
"name": "buyram",
```

```
"base": "",
```

```
"fields": [{
```

```
    "name": "payer",
    "type": "name"
  }, {
    "name": "receiver",
    "type": "name"
  }, {
    "name": "quant",
    "type": "asset"
  }
]
}, {
  "name": "buyrambytes",
  "base": "",
  "fields": [ {
    "name": "payer",
    "type": "name"
  }, {
    "name": "receiver",
```

```
    "type": "name"
  }, {
    "name": "bytes",
    "type": "uint32"
  }
]
}, {
  "name": "canceledelay",
  "base": "",
  "fields": [ {
    "name": "canceling_auth",
    "type": "permission_level"
  }, {
    "name": "trx_id",
    "type": "checksum256"
  }
]
}, {
```

```
"name": "claimrewards",
"base": "",
"fields": [{
  "name": "owner",
  "type": "name"
}]
}, {
"name": "connector",
"base": "",
"fields": [{
  "name": "balance",
  "type": "asset"
}], {
  "name": "weight",
  "type": "float64"
}
]
```

```
}, {  
  "name": "delegatebw",  
  "base": "",  
  "fields": [{  
    "name": "from",  
    "type": "name"  
  }, {  
    "name": "receiver",  
    "type": "name"  
  }, {  
    "name": "stake_net_quantity",  
    "type": "asset"  
  }, {  
    "name": "stake_cpu_quantity",  
    "type": "asset"  
  }, {  
    "name": "transfer",  
    "type": "bool"
```



```
    }  
  ]  
}, {  
  "name": "delegated_bandwidth",  
  "base": "",  
  "fields": [{  
    "name": "from",  
    "type": "name"  
  }, {  
    "name": "to",  
    "type": "name"  
  }, {  
    "name": "net_weight",  
    "type": "asset"  
  }, {  
    "name": "cpu_weight",  
    "type": "asset"  
  }  
}
```

```
]
}, {
  "name": "deleteauth",
  "base": "",
  "fields": [ {
    "name": "account",
    "type": "name"
  }, {
    "name": "permission",
    "type": "name"
  }
]
}, {
  "name": "eosio_global_state",
  "base": "blockchain_parameters",
  "fields": [ {
    "name": "max_ram_size",
    "type": "uint64"
```

```
}, {
```

```
"name":
```

```
"total_ram_bytes_reserved",
```

```
"type": "uint64"
```

```
}, {
```

```
"name": "total_ram_stake",
```

```
"type": "int64"
```

```
}, {
```

```
"name":
```

```
"last_producer_schedule_update",
```

```
"type": "block_timestamp_type"
```

```
}, {
```

```
"name":
```

```
"last_pervote_bucket_fill",
```

```
"type": "time_point"
```

```
}, {
```

```
"name": "pervote_bucket",
```

```
"type": "int64"
```

```
}, {
  "name": "perblock_bucket",
  "type": "int64"
}, {
  "name": "total_unpaid_blocks",
  "type": "uint32"
}, {
  "name": "total_activated_stake",
  "type": "int64"
}, {
```

"name":

```
"thresh_activated_stake_time",
  "type": "time_point"
}, {
```

"name":

```
"last_producer_schedule_size",
  "type": "uint16"
}, {
```

"name":

```
"total_producer_vote_weight",
  "type": "float64"
}, {
  "name": "last_name_close",
  "type": "block_timestamp_type"
}
]
}, {
  "name": "eosio_global_state2",
  "base": "",
  "fields": [{
    "name": "new_ram_per_block",
    "type": "uint16"
  }, {
    "name": "last_ram_increase",
    "type": "block_timestamp_type"
  }, {
```

```
    "name": "last_block_num",
    "type": "block_timestamp_type"
  }, {
    "name":
"total_producer_votepay_share",
    "type": "float64"
  }, {
    "name": "revision",
    "type": "uint8"
  }
]
}, {
  "name": "eosio_global_state3",
  "base": "",
  "fields": [{
    "name": "last_vpay_state_update",
    "type": "time_point"
  }, {
```

"name":

"total_vpay_share_change_rate",

"type": "float64"

}

]

},{

"name": "exchange_state",

"base": "",

"fields": [{

"name": "supply",

"type": "asset"

},{

"name": "base",

"type": "connector"

},{

"name": "quote",

"type": "connector"

}

```
]
}, {
  "name": "init",
  "base": "",
  "fields": [{
    "name": "version",
    "type": "varuint32"
  }, {
    "name": "core",
    "type": "symbol"
  }
]
}, {
  "name": "key_weight",
  "base": "",
  "fields": [{
    "name": "key",
    "type": "public_key"
```



```
    }, {  
      "name": "weight",  
      "type": "uint16"  
    }  
  ]  
}, {  
  "name": "linkauth",  
  "base": "",  
  "fields": [{  
    "name": "account",  
    "type": "name"  
  }, {  
    "name": "code",  
    "type": "name"  
  }, {  
    "name": "type",  
    "type": "name"  
  }  
}, {
```

```
    "name": "requirement",
    "type": "name"
  }
]
}, {
  "name": "name_bid",
  "base": "",
  "fields": [{
    "name": "newname",
    "type": "name"
  }, {
    "name": "high_bidder",
    "type": "name"
  }, {
    "name": "high_bid",
    "type": "int64"
  }, {
    "name": "last_bid_time",
```

```
    "type": "time_point"
  }
]
}, {
  "name": "newaccount",
  "base": "",
  "fields": [{
    "name": "creator",
    "type": "name"
  }, {
    "name": "name",
    "type": "name"
  }, {
    "name": "owner",
    "type": "authority"
  }, {
    "name": "active",
    "type": "authority"
  }
]
```

```
    }  
  ]  
}, {  
  "name": "onblock",  
  "base": "",  
  "fields": [{  
    "name": "header",  
    "type": "block_header"  
  }  
]  
}, {  
  "name": "onerror",  
  "base": "",  
  "fields": [{  
    "name": "sender_id",  
    "type": "uint128"  
  }], {  
    "name": "sent_trx",
```

```
    "type": "bytes"
  }
]
}, {
  "name": "permission_level",
  "base": "",
  "fields": [{
    "name": "actor",
    "type": "name"
  }, {
    "name": "permission",
    "type": "name"
  }
]
}, {
  "name": "permission_level_weight",
  "base": "",
  "fields": [{
```

```
    "name": "permission",
    "type": "permission_level"
  }, {
    "name": "weight",
    "type": "uint16"
  }
]
}, {
  "name": "producer_info",
  "base": "",
  "fields": [ {
    "name": "owner",
    "type": "name"
  }, {
    "name": "total_votes",
    "type": "float64"
  }, {
    "name": "producer_key",
```

```
    "type": "public_key"
  }, {
    "name": "is_active",
    "type": "bool"
  }, {
    "name": "url",
    "type": "string"
  }, {
    "name": "unpaid_blocks",
    "type": "uint32"
  }, {
    "name": "last_claim_time",
    "type": "time_point"
  }, {
    "name": "location",
    "type": "uint16"
  }
]
```

```
}, {  
  "name": "producer_info2",  
  "base": "",  
  "fields": [{  
    "name": "owner",  
    "type": "name"  
  }], {  
    "name": "votepay_share",  
    "type": "float64"  
  }], {
```

"name":

```
"last_votepay_share_update",  
  "type": "time_point"  
}  
]  
}, {  
  "name": "producer_key",  
  "base": "",
```



```
"fields": [{
  "name": "producer_name",
  "type": "name"
}, {
  "name": "block_signing_key",
  "type": "public_key"
}
]
}, {
  "name": "producer_schedule",
  "base": "",
  "fields": [{
    "name": "version",
    "type": "uint32"
  }, {
    "name": "producers",
    "type": "producer_key[]"
  }
}
```

```
]
}, {
  "name": "refund",
  "base": "",
  "fields": [{
    "name": "owner",
    "type": "name"
  }]
}, {
  "name": "refund_request",
  "base": "",
  "fields": [{
    "name": "owner",
    "type": "name"
  }], {
  "name": "request_time",
  "type": "time_point_sec"
}
```

```
    }, {
      "name": "net_amount",
      "type": "asset"
    }, {
      "name": "cpu_amount",
      "type": "asset"
    }
  ]
}, {
  "name": "regproducer",
  "base": "",
  "fields": [ {
    "name": "producer",
    "type": "name"
  }, {
    "name": "producer_key",
    "type": "public_key"
  }, {
```

```
    "name": "url",
    "type": "string"
  }, {
    "name": "location",
    "type": "uint16"
  }
]
}, {
  "name": "regproxy",
  "base": "",
  "fields": [ {
    "name": "proxy",
    "type": "name"
  }, {
    "name": "isproxy",
    "type": "bool"
  }
]
]
```

```
}, {  
  "name": "rmvproducer",  
  "base": "",  
  "fields": [{  
    "name": "producer",  
    "type": "name"  
  }  
]
```

```
}, {  
  "name": "sellram",  
  "base": "",  
  "fields": [{  
    "name": "account",  
    "type": "name"  
  }], {  
    "name": "bytes",  
    "type": "int64"  
  }  
}
```

```
]
}, {
  "name": "setabi",
  "base": "",
  "fields": [{
    "name": "account",
    "type": "name"
  }, {
    "name": "abi",
    "type": "bytes"
  }
]
}, {
  "name": "setacctcpu",
  "base": "",
  "fields": [{
    "name": "account",
    "type": "name"
```

```
    }, {
      "name": "cpu_weight",
      "type": "int64?"
    }
  ]
}, {
  "name": "setacctnet",
  "base": "",
  "fields": [ {
    "name": "account",
    "type": "name"
  }, {
    "name": "net_weight",
    "type": "int64?"
  }
  ]
}, {
  "name": "setacctram",
```

```
"base": "",
"fields": [{
  "name": "account",
  "type": "name"
}, {
  "name": "ram_bytes",
  "type": "int64?"
}
]
}, {
  "name": "setalimits",
  "base": "",
  "fields": [{
    "name": "account",
    "type": "name"
  }, {
    "name": "ram_bytes",
    "type": "int64"
```



```
    }, {
      "name": "net_weight",
      "type": "int64"
    }, {
      "name": "cpu_weight",
      "type": "int64"
    }
  ]
}, {
  "name": "setcode",
  "base": "",
  "fields": [ {
    "name": "account",
    "type": "name"
  }, {
    "name": "vmtype",
    "type": "uint8"
  }, {
```

```
    "name": "vmversion",
    "type": "uint8"
  }, {
    "name": "code",
    "type": "bytes"
  }
]
}, {
  "name": "setparams",
  "base": "",
  "fields": [{
    "name": "params",
    "type": "blockchain_parameters"
  }
]
}, {
  "name": "setpriv",
  "base": "",
```

```
"fields": [{
  "name": "account",
  "type": "name"
}, {
  "name": "is_priv",
  "type": "uint8"
}
]
}, {
  "name": "setram",
  "base": "",
  "fields": [{
    "name": "max_ram_size",
    "type": "uint64"
  }
]
}, {
  "name": "setramrate",
```

```
"base": "",
"fields": [{
  "name": "bytes_per_block",
  "type": "uint16"
}]
}, {
  "name": "undelegatebw",
  "base": "",
  "fields": [{
    "name": "from",
    "type": "name"
  }, {
    "name": "receiver",
    "type": "name"
  }, {
    "name": "unstake_net_quantity",
    "type": "asset"
```

```
    }, {
      "name": "unstake_cpu_quantity",
      "type": "asset"
    }
  ]
}, {
  "name": "unlinkauth",
  "base": "",
  "fields": [ {
    "name": "account",
    "type": "name"
  }, {
    "name": "code",
    "type": "name"
  }, {
    "name": "type",
    "type": "name"
  }
}
```

```
]
}, {
  "name": "unregprod",
  "base": "",
  "fields": [{
    "name": "producer",
    "type": "name"
  }]
}]
}, {
  "name": "updateauth",
  "base": "",
  "fields": [{
    "name": "account",
    "type": "name"
  }], {
  "name": "permission",
  "type": "name"
}
```

```
    }, {
      "name": "parent",
      "type": "name"
    }, {
      "name": "auth",
      "type": "authority"
    }
  ]
}, {
  "name": "updtrevision",
  "base": "",
  "fields": [ {
    "name": "revision",
    "type": "uint8"
  }
  ]
}, {
  "name": "user_resources",
```

```
"base": "",
"fields": [{
  "name": "owner",
  "type": "name"
}, {
  "name": "net_weight",
  "type": "asset"
}, {
  "name": "cpu_weight",
  "type": "asset"
}, {
  "name": "ram_bytes",
  "type": "int64"
}
]
}, {
  "name": "voteproducer",
  "base": "",
```



```
"fields": [{
  "name": "voter",
  "type": "name"
}, {
  "name": "proxy",
  "type": "name"
}, {
  "name": "producers",
  "type": "name[]"
}
]
}, {
  "name": "voter_info",
  "base": "",
  "fields": [{
    "name": "owner",
    "type": "name"
  }, {
```

```
"name": "proxy",
"type": "name"
}, {
  "name": "producers",
  "type": "name[]"
}, {
  "name": "staked",
  "type": "int64"
}, {
  "name": "last_vote_weight",
  "type": "float64"
}, {
  "name": "proxied_vote_weight",
  "type": "float64"
}, {
  "name": "is_proxy",
  "type": "bool"
}, {
```

```
    "name": "flags1",
    "type": "uint32"
  }, {
    "name": "reserved2",
    "type": "uint32"
  }, {
    "name": "reserved3",
    "type": "asset"
  }
]
}, {
  "name": "wait_weight",
  "base": "",
  "fields": [ {
    "name": "wait_sec",
    "type": "uint32"
  }, {
    "name": "weight",
```

```
"type": "uint16"
```

```
}
```

```
]
```

```
}
```

```
],
```

```
"actions": [{
```

```
  "name": "bidname",
```

```
  "type": "bidname",
```

```
  "ricardian_contract": "# Action - `{{
```

```
bidname }}`\n\n### Description\n\nThe
```

```
`{{ bidname }}` action places a bid on a
```

```
premium account name, in the
```

```
knowledge that the high bid will
```

```
purchase the name.\n\nAs an authorized
```

```
party I {{ signer }} wish to bid on
```

```
behalf of {{ bidder }} the amount of {{
```

```
bid }} toward purchase of the account
```

```
name {{ newname }}.\n"
```

```
}, {  
  "name": "bidrefund",  
  "type": "bidrefund",  
  "ricardian_contract": ""
```

```
}, {  
  "name": "buyram",  
  "type": "buyram",  
  "ricardian_contract": "# Action - `{{
```

```
buyram }}`  
\\n\\n#### Description\\n\\nThis  
action will attempt to reserve about  
{{quant}} worth of RAM on behalf of  
{{receiver}}. \\n\\n{{buyer}} authorizes  
this contract to transfer {{quant}} to buy  
RAM based upon the current price as  
determined by the market maker  
algorithm.\\n\\n{{buyer}} accepts that a  
0.5% fee will be charged on the amount  
spent and that the actual RAM received
```

may be slightly less than expected due to the approximations necessary to enable this service. \n{ {buyer} } accepts that a 0.5% fee will be charged if and when they sell the RAM received. \n{ {buyer} } accepts that rounding errors resulting from limits of computational precision may result in less RAM being allocated. \n{ {buyer} } acknowledges that the supply of RAM may be increased at any time up to the limits of off-the-shelf computer equipment and that this may result in RAM selling for less than purchase price. \n{ {buyer} } acknowledges that the price of RAM may increase or decrease over time according to supply and demand. \n{ {buyer} } acknowledges that

RAM is non-transferrable. `{buyer}` acknowledges RAM currently in use by their account cannot be sold until it is freed and that freeing RAM may be subject to terms of other contracts.

}, {

"name": "buyrambytes",

"type": "buyrambytes",

"ricardian_contract": "# Action - `{{ buyrambytes }}`\n\n###

Description\n\nThis action will attempt to reserve about `{bytes}` bytes of RAM on behalf of `{receiver}`.
`{buyer}` authorizes this contract to transfer sufficient EOS tokens to buy the RAM based upon the current price as determined by the market maker algorithm.
`{buyer}` accepts that a

0.5% fee will be charged on the EOS spent and that the actual RAM received may be slightly less than requested due to the approximations necessary to enable this service. \n{ {buyer} } accepts that a 0.5% fee will be charged if and when they sell the RAM received. \n{ {buyer} } accepts that rounding errors resulting from limits of computational precision may result in less RAM being allocated. \n{ {buyer} } acknowledges that the supply of RAM may be increased at any time up to the limits of off-the-shelf computer equipment and that this may result in RAM selling for less than purchase price. \n{ {buyer} } acknowledges that the price of RAM may increase or decrease

over time according to supply and demand.\n{{buyer}} acknowledges that RAM is non-transferable. \n{{buyer}} acknowledges RAM currently in use by their account cannot be sold until it is freed and that freeing RAM may be subject to terms of other contracts.\n\n"

},{

"name": "canceledelay",

"type": "canceledelay",

"ricardian_contract": "# Action - `{{ canceledelay`"}}

Description\n\nThe `{{ canceledelay` action cancels an existing delayed transaction.\n\nAs an authorized party I {{ signer }} wish to invoke the authority of {{ canceling_auth }} to cancel the transaction with ID {{ trx_id }}.\n"

```
}, {
```

```
  "name": "claimrewards",
```

```
  "type": "claimrewards",
```

```
  "ricardian_contract": "# Action - `{{  
claimrewards      }}`\n\n###
```

```
Description\n\nThe `{{ claimrewards  
}}` action allows a block producer  
(active or standby) to claim the system  
rewards due them for producing blocks  
and receiving votes.\n\nAs an authorized  
party I {{ signer }} wish to have the  
rewards earned by {{ owner }}  
deposited into the {{ owner }}  
account.\n"
```

```
}, {
```

```
  "name": "delegatebw",
```

```
  "type": "delegatebw",
```

```
  "ricardian_contract": "# Action - `{{
```

delegatebw } }` \n\n###

Description\n\nThe intent of the `{{ delegatebw }}` action is to stake tokens for bandwidth and/or CPU and optionally transfer ownership.\n\nAs an authorized party I {{ signer }} wish to stake {{ stake_cpu_quantity }} for CPU and {{ stake_net_quantity }} for bandwidth from the liquid tokens of {{ from }} for the use of delegatee {{ to }}. \n \n {{if transfer }}\n \nIt is {{ transfer }} that I wish these tokens to become immediately owned by the delegatee.\n \n {{/if}}\n\nAs signer I stipulate that, if I am not the beneficial owner of these tokens, I have proof that Iu2019ve been authorized to take this action by their beneficial owner(s). \n"

```
}, {  
  "name": "deleteauth",  
  "type": "deleteauth",  
  "ricardian_contract": ""
```

```
}, {  
  "name": "init",  
  "type": "init",  
  "ricardian_contract": ""
```

```
}, {  
  "name": "linkauth",  
  "type": "linkauth",  
  "ricardian_contract": ""
```

```
}, {  
  "name": "newaccount",  
  "type": "newaccount",  
  "ricardian_contract": "# Action - `{{  
newaccount                                `}}`\n\n###`
```

```
Description\n\nThe `{{ newaccount `}}`
```

action creates a new account.\n\nAs an authorized party I {{ signer }} wish to exercise the authority of {{ creator }} to create a new account on this system named {{ name }} such that the new account's owner public key shall be {{ owner }} and the active public key shall be {{ active }}.\n"

},{

"name": "onblock",

"type": "onblock",

"ricardian_contract": ""

},{

"name": "onerror",

"type": "onerror",

"ricardian_contract": ""

},{

"name": "refund",

"type": "refund",

"ricardian_contract": "# Action - `{{

refund }}`\n\n### Description\n\nThe intent of the `{{ refund }}` action is to return previously unstaked tokens to an account after the unstaking period has elapsed. \n\nAs an authorized party I {{ signer }} wish to have the unstaked tokens of {{ owner }} returned.\n"

}, {

"name": "regproducer",

"type": "regproducer",

"ricardian_contract": "# Action - `{{

regproducer }}`\n\n### Description\n\nThe intent of the `{{ regproducer }}` action is to register an account as a BP candidate.\n\nI, {{ producer }}, hereby nominate myself

for consideration as an elected block producer.\n\nIf {{producer}} is selected to produce blocks by the eosio contract, I will sign blocks with {{producer_key}} and I hereby attest that I will keep this key secret and secure.\n\nIf {{producer}} is unable to perform obligations under this contract I will resign my position by resubmitting this contract with the null producer key.\n\nI acknowledge that a block is 'objectively valid' if it conforms to the deterministic blockchain rules in force at the time of its creation, and is 'objectively invalid' if it fails to conform to those rules.\n\n{{producer}} hereby agrees to only use {{producer_key}} to sign messages under the following

scenarios:\nproposing an objectively valid block at the time appointed by the block scheduling algorithm\npre-confirming a block produced by another producer in the schedule when I find said block objectively valid\nconfirming a block for which {{producer}} has received 2/3+ pre-confirmation messages from other producers\n\nI hereby accept liability for any and all provable damages that result from my:\n\nsigning two different block proposals with the same timestamp with {{producer_key}}\n\nsigning two different block proposals with the same block number with {{producer_key}}\n\nsigning any block proposal which builds off of an

objectively invalid block\nsigning a pre-confirmation for an objectively invalid block\nsigning a confirmation for a block for which I do not possess pre-confirmation messages from 2/3+ other producers\n\nI hereby agree that double-signing for a timestamp or block number in concert with 2 or more other producers shall automatically be deemed malicious and subject to a fine equal to the past year of compensation received and immediate disqualification from being a producer, and other damages. An exception may be made if {{producer}} can demonstrate that the double-signing occurred due to a bug in the reference software; the burden of proof is on {{producer}}.\n\nI hereby agree not to

interfere with the producer election process. I agree to process all producer election transactions that occur in blocks I create, to sign all objectively valid blocks I create that contain election transactions, and to sign all pre-confirmations and confirmations necessary to facilitate transfer of control to the next set of producers as determined by the system contract.

I hereby acknowledge that 2/3+ other elected producers may vote to disqualify {{producer}} in the event {{producer}} is unable to produce blocks or is unable to be reached, according to criteria agreed to among producers.

If {{producer}} qualifies for and chooses to collect compensation due to votes

received, {{producer}} will provide a public endpoint allowing at least 100 peers to maintain synchronization with the blockchain and/or submit transactions to be included. {{producer}} shall maintain at least 1 validating node with full state and signature checking and shall report any objectively invalid blocks produced by the active block producers. Reporting shall be via a method to be agreed to among producers, said method and reports to be made public.\n\nThe community agrees to allow {{producer}} to authenticate peers as necessary to prevent abuse and denial of service attacks; however, {{producer}} agrees not to discriminate against non-

abusive peers.\n\nI agree to process transactions on a FIFO best-effort basis and to honestly bill transactions for measured execution time.\n\n{{producer}} agree not to manipulate the contents of blocks in order to derive profit from:\n\nthe order in which transactions are included\n\nthe hash of the block that is produced\n\nI, {{producer}}, hereby agree to disclose and attest under penalty of perjury all ultimate beneficial owners of my company who own more than 10% and all direct shareholders.\n\nI, {{producer}}, hereby agree to cooperate with other block producers to carry out our respective and mutual obligations under this agreement,

including but not limited to maintaining network stability and a valid blockchain.

I, {{producer}}, agree to maintain a website hosted at {{url}} which contains up-to-date information on all disclosures required by this contract.

I, {{producer}}, agree to set {{location}} such that {{producer}} is scheduled with minimal latency between my previous and next peer.

I, {{producer}}, agree to maintain time synchronization within 10 ms of global atomic clock time, using a method agreed to among producers.

I, {{producer}}, agree not to produce blocks before my scheduled time unless I have received all blocks produced by the prior producer.

I, {{producer}},

agree not to publish blocks with timestamps more than 500ms in the future unless the prior block is more than 75% full by either CPU or network bandwidth metrics.\n\nI, {{producer}}, agree not to set the RAM supply to more RAM than my nodes contain and to resign if I am unable to provide the RAM approved by 2/3+ producers, as shown in the system parameters.\n"

},{

"name": "regproxy",

"type": "regproxy",

"ricardian_contract": ""

},{

"name": "rmvproducer",

"type": "rmvproducer",

"ricardian_contract": ""

```
}, {  
  "name": "sellram",  
  "type": "sellram",  
  "ricardian_contract": "# Action - `{{  
sellram }}`\n\n### Description\n\nThe  
`{{ sellram }}` action sells unused  
RAM for tokens.\n\nAs an authorized  
party I {{ signer }} wish to sell {{ bytes  
}} of unused RAM from account {{  
account }}. \n"
```

```
}, {  
  "name": "setabi",  
  "type": "setabi",  
  "ricardian_contract": ""
```

```
}, {  
  "name": "setacctcpu",  
  "type": "setacctcpu",  
  "ricardian_contract": ""
```

```
}, {  
  "name": "setacctnet",  
  "type": "setacctnet",  
  "ricardian_contract": ""  
}, {  
  "name": "setacctram",  
  "type": "setacctram",  
  "ricardian_contract": ""  
}, {  
  "name": "setalimits",  
  "type": "setalimits",  
  "ricardian_contract": ""  
}, {  
  "name": "setcode",  
  "type": "setcode",  
  "ricardian_contract": ""  
}, {  
  "name": "setparams",
```



```
"type": "setparams",  
"ricardian_contract": ""  
}, {  
  "name": "setpriv",  
  "type": "setpriv",  
  "ricardian_contract": ""  
}, {  
  "name": "setram",  
  "type": "setram",  
  "ricardian_contract": ""  
}, {  
  "name": "setramrate",  
  "type": "setramrate",  
  "ricardian_contract": ""  
}, {  
  "name": "undelegatebw",  
  "type": "undelegatebw",  
  "ricardian_contract": "# Action - `{{
```

undelegatebw } }` \n\n##

Description\n\nThe intent of the `{{ undelegatebw }}` action is to un stake tokens from CPU and/or bandwidth.

\n\nAs an authorized party I {{ signer }} wish to un stake {{ un stake_cpu_quantity }} from CPU and {{ un stake_net_quantity }} from bandwidth from the tokens owned by {{ from }} previously delegated for the use of delegatee {{ to }}. \n\nIf I as signer am not the beneficial owner of these tokens I stipulate I have proof that I've been authorized to take this action by their beneficial owner(s). \n"

},{

"name": "unlinkauth",

"type": "unlinkauth",

```
"ricardian_contract": ""
```

```
}, {
```

```
  "name": "unregprod",
```

```
  "type": "unregprod",
```

```
  "ricardian_contract": "# Action - `{{
```

```
unregprod }}`\n\n### Description\n\nThe
```

```
`{{ unregprod }}` action unregisters a
```

```
previously registered block producer
```

```
candidate.\n\nAs an authorized party I {{
```

```
signer }} wish to unregister the block
```

```
producer candidate {{ producer }},
```

```
rendering that candidate no longer able
```

```
to receive votes.\n"
```

```
}, {
```

```
  "name": "updateauth",
```

```
  "type": "updateauth",
```

```
  "ricardian_contract": ""
```

```
}, {
```

```
"name": "updtrevision",
"type": "updtrevision",
"ricardian_contract": ""
```

```
}, {
```

```
"name": "voteproducer",
"type": "voteproducer",
```

```
"ricardian_contract": "# Action - `{{`  
voteproducer`}}`\n\n###
```

```
Description\n\nThe intent of the `{{`  
voteproducer`}}` action is to cast a  
valid vote for up to 30 BP candidates.  
\n\nAs an authorized party I {{ signer }}  
wish to vote on behalf of {{ voter }} in  
favor of the block producer candidates  
{{ producers }} with a voting weight  
equal to all tokens currently owned by  
{{ voter }} and staked for CPU or  
bandwidth. \n\nIf I am not the beneficial
```

owner of these shares I stipulate I have proof that I've been authorized to vote these shares by their beneficial owner(s). \n\nI stipulate I have not and will not accept anything of value in exchange for these votes, on penalty of confiscation of these tokens, and other penalties. \n\nI acknowledge that using any system of automatic voting, re-voting, or vote refreshing, or allowing such a system to be used on my behalf or on behalf of another, is forbidden and doing so violates this contract.\n"

}

],

```
"tables": [{  
  "name": "abihash",  
  "index_type": "i64",
```

```
"key_names": [],  
"key_types": [],  
"type": "abi_hash"  
}, {  
  "name": "bidrefunds",  
  "index_type": "i64",  
  "key_names": [],  
  "key_types": [],  
  "type": "bid_refund"  
}, {  
  "name": "delband",  
  "index_type": "i64",  
  "key_names": [],  
  "key_types": [],  
  "type": "delegated_bandwidth"  
}, {  
  "name": "global",  
  "index_type": "i64",
```

```
"key_names": [],  
"key_types": [],  
"type": "eosio_global_state"  
}, {  
  "name": "global2",  
  "index_type": "i64",  
  "key_names": [],  
  "key_types": [],  
  "type": "eosio_global_state2"  
}, {  
  "name": "global3",  
  "index_type": "i64",  
  "key_names": [],  
  "key_types": [],  
  "type": "eosio_global_state3"  
}, {  
  "name": "namebids",  
  "index_type": "i64",
```

```
"key_names": [],  
"key_types": [],  
"type": "name_bid"  
}, {  
  "name": "producers",  
  "index_type": "i64",  
  "key_names": [],  
  "key_types": [],  
  "type": "producer_info"  
}, {  
  "name": "producers2",  
  "index_type": "i64",  
  "key_names": [],  
  "key_types": [],  
  "type": "producer_info2"  
}, {  
  "name": "rammarket",  
  "index_type": "i64",
```



```
"key_names": [],  
"key_types": [],  
"type": "exchange_state"  
}, {  
  "name": "refunds",  
  "index_type": "i64",  
  "key_names": [],  
  "key_types": [],  
  "type": "refund_request"  
}, {  
  "name": "userres",  
  "index_type": "i64",  
  "key_names": [],  
  "key_types": [],  
  "type": "user_resources"  
}, {  
  "name": "voters",  
  "index_type": "i64",
```

```
"key_names": [],  
"key_types": [],  
"type": "voter_info"
```

```
}
```

```
],
```

```
"ricardian_clauses": [{  
  "id": "constitution",
```

```
    "body": "This Constitution is a  
multi-party contract entered into by the  
Members by virtue of their use of this  
blockchain.\n\n# Article I - No Initiation  
of Violence\nMembers shall not initiate  
violence or the threat of violence against  
another Member. Lawful prosecution of  
crimes with the goal of preserving life,  
liberty and property does not constitute  
initiation of violence.\n\n# Article II -  
No Perjury\nMembers shall be liable for
```

losses caused by false or misleading attestations and shall forfeit any profit gained thereby.

Article III - Rights
The Members grant the right of contract and of private property to each other, therefore no property shall change hands except with the consent of the owner, by a valid Arbitrator's order, or via community referendum. This Constitution creates no positive rights for or between any Members.

Article IV - No Vote Buying
No Member shall offer nor accept anything of value in exchange for a vote of any type, nor shall any Member unduly influence the vote of another.

Article V - No Fiduciary
No Member nor EOS token holder shall have

fiduciary responsibility to support the value of the EOS token. The Members do not authorize anyone to hold assets, borrow, nor contract on behalf of EOS token holders collectively. This blockchain has no owners, managers or fiduciaries; therefore, no Member shall have beneficial interest in more than 10% of the EOS token supply.

Article VI - Restitution
Each Member agrees that penalties for breach of contract may include, but are not limited to, fines, loss of account, and other restitution.

Article VII - Open Source
Each Member who makes available a smart contract on this blockchain shall be a Developer. Each Developer shall offer their smart

contracts via a free and open source license, and each smart contract shall be documented with a Ricardian Contract stating the intent of all parties and naming the Arbitration Forum that will resolve disputes arising from that contract.

Article VIII - Language
Multi-lingual contracts must specify one prevailing language in case of dispute and the author of any translation shall be liable for losses due to their false, misleading, or ambiguous attested translations.

Article IX - Dispute Resolution
All disputes arising out of or in connection with this Constitution shall be finally settled under the Rules of Dispute Resolution of the EOS Core Arbitration Forum by one

or more arbitrators appointed in accordance with the said Rules.

Article X - Choice of Law
Choice of law for disputes shall be, in order of precedence, this Constitution and the Maxims of Equity.

Article XI - Amending
This Constitution and its subordinate documents shall not be amended except by a vote of the token holders with no less than 15% vote participation among tokens and no fewer than 10% more Yes than No votes, sustained for 30 continuous days within a 120 day period.

Article XII - Publishing
Members may only publish information to the Blockchain that is within their right to publish. Furthermore, Members voluntarily

consent for all Members to permanently and irrevocably retain a copy, analyze, and distribute all broadcast transactions and derivative information.

Article XIII - Informed Consent

All service providers who produce tools to facilitate the construction and signing of transactions on behalf of other Members shall present to said other Members the full Ricardian contract terms of this Constitution and other referenced contracts. Service providers shall be liable for losses resulting from failure to disclose the full Ricardian contract terms to users.

Article XIV - Severability

If any part of this Constitution is declared unenforceable or invalid, the remainder will continue

to be valid and enforceable.\n\n# Article XV - Termination of Agreement\nA Member is automatically released from all revocable obligations under this Constitution 3 years after the last transaction signed by that Member is incorporated into the blockchain. After 3 years of inactivity an account may be put up for auction and the proceeds distributed to all Members according to the system contract provisions then in effect for such redistribution.\n\n#

Article XVI - Developer Liability\nMembers agree to hold software developers harmless for unintentional mistakes made in the expression of contractual intent, whether or not said mistakes were due to actual

or perceived negligence.\n\n# Article XVII - Consideration\nAll rights and obligations under this Constitution are mutual and reciprocal and of equally significant value and cost to all parties.\n\n# Article XVIII - Acceptance\nA contract is deemed accepted when a member signs a transaction which incorporates a TAPOS proof of a block whose implied state incorporates an ABI of said contract and said transaction is incorporated into the blockchain.\n\n# Article XIX - Counterparts\nThis Constitution may be executed in any number of counterparts, each of which when executed and delivered shall constitute a duplicate original, but all

counterparts together shall constitute a single agreement.\n\n# Article XX - Interim Constitution\n\nThis constitution is interim and is intended to remain in effect until a permanent constitution is written and ratified in a referendum.\n"

}

],

"error_messages": [],

"abi_extensions": [],

"variants": []

}

- get schedule

Recupera il programma del producer.

Positionals:

none

Options:

-h, --help - stampa un messaggio di aiuto ed esce

-j, --json - l'output verrà in formato json

esempio:

```
cleos get schedule
```

```
active schedule version 695
```

```
  Producer      Producer key
```

```
=====
```

```
=====
```

```
atticlabeosb
```

EOS7PfA3A4UdfMu2wKbuXdbHn8EW.
bitfinexeos1

EOS4tkw7LgtURT3dvG3kQ4D1sg3aAtP
cochainworld

EOS5QDSQyh96SmktA28dteEchc1QCV
eos42freedom

EOS4tw7vH62TcVtMgm2tjXzn9hTuHEI
eosauthority

EOS4va3CTmAcAAXsT26T3EBWqYH
eosbeijingbp

EOS5dGpcEhwB4VEhhXEcqtZs9EQj5H
eosbixinboot

EOS7QC1XfAtkYeLjbHQjcDWVqUsxu6
eoscanadacom

EOS7eycxAbCtKyfoJc8uRZcmt1AmArj2
eos cannonchn

EOS73cTi9V7PNg4ujW5QzoTfRSdhH4
eosflytomars

EOS6Agpfp38bTyRjJDmB4Qb1EpQSq7
eoshenzhenio

EOS8EJrMphgHJx5EkHQ4ryodbvnocZE
eoshuobipool

EOS5XKswW26cR5VQeDGwgNb5aixv
eosiosg11111

EOS7zVBQMhV7dZ5zRQwBgDmmbFC
eoslaomaocom

EOS8QgURqo875qu3a8vgZ58qBeu2cTe
eosliquideos

EOS4v1n2j5kXbCum8LLEc8zQLpeLK91
eosnationftw

EOS5x4RBDk7ekzKp8ixsntAxWoQjMG
eosnewyorkio

EOS6GVX8eUqC1gN1293B3ivCNbifbr
eosriobrazil

EOS7RioGoHQnhv2fJEiciP9Q7J8JgfJY1
jedaaaaaaa

EOS6nB9Ar5sghWjqk27bszCiA9zxQtXZ
starteosiobp
EOS4wZZXm994byKANLuwHD6tV3R3
zbeosbp11111
EOS7rhgVPWWyflMqjSbNdndtCK8Gkza

pending schedule empty

proposed schedule empty

- get transactions_id

Positionals:

transactions TEXT - la stringa json o il filename definito della transazione da dove vogliamo recuperare l'ed della

transazione, il file json deve contenere le transazioni impacchettate per funzionare correttamente

Options:

-h, --help - stampa un messaggio di aiuto ed esce

Usage:

```
cleos get transaction_id ./ptrx.json
```

Output:

```
374708fff7719dd5979ec875d56cd2286f
```

- set contract

Positionals:

account TEXT - l'account per pubblicare il contratto

contract-dir TEXT - il percorso contenente il file .wast e .abi

wast-file TEXT - il file contenente il contratto o il file -wast

abi-file TEXT - il file .abi per il contratto

Options:

-h, --help - stampa un messaggio di aiuto ed esce

-a, --abi TEXT - l'ABI per il contratto

-x, --expiration TEXT - setta il tempo in secondi prima che il contratto finisca, di

default è 30 secondi

-f, --force-unique - forza la transazione ad essere unica, questo consumerà banda extra e rimuoverà qualsiasi protezione contro errori multipli provenienti dalla stessa transazione

-s, --skip-sign - specifica se un wallet sbloccato può essere usato per firmare la transazione

-d, --dont-broadcast - non trasmette la transazione alla rete (la stampa al stdout)

-r, --ref-block TEXT - setta i riferimenti del numero del blocco o l'id del blocco usato per la TAPOS

-p, --permission TEXT - un account ed un livello di permesso, come in account@permission (default è

account@active)

--max-cpu-usage-ms UINT - setta un tetto massimo di millisecondi per l'utilizzo della cpu, per l'esecuzione del contratto (default è 0 cioè senza limiti)

--max-net-usage UINT - setta un tetto massimo di net da usare, in bytes, per la transazione (default è 0 cioè non ci sono limiti)

Usage:

un contratto di una moneta tipica

```
$ ./cleos set contract currency
```

```
../../../../contracts/currency/currency.wast
```

```
../../../../contracts/currency/currency.abi
```

Output:

Reading WAST...

Assembling WASM...

Publishing contract...

```
{
    "transaction_id":
"9990306e13f630a9c5436a5a0b6fb8fe2c
"processed": {
    "refBlockNum": 1208,
    "refBlockPrefix": 3058534156,
    "expiration": "2017-08-
24T18:29:52",
    "scope": [
        "currency",
        "eos"
    ],
    "signatures": [],
    "messages": [ {
```

```
"code": "eos",  
"type": "setcode",  
"authorization": [{  
  "account": "currency",  
  "permission": "active"  
}]  
],
```

```
"data":
```

```
"00000079b822651d0000e8150061736c"  
}
```

```
],
```

```
"output": [{  
  "notify": [],  
  "sync_transactions": [],  
  "async_transactions": []  
}]
```

```
]
```

```
}
```

}

- set code

Positionals:

account TEXT - l'account per settare il

codice

code-file TEXT - il percorso completo che contiene il contratto o WAST o WASM (richiesto)

Options:

-h, --help - stampa un messaggio di aiuto ed esce

-a, --abi TEXT - l'ABI per il contratto

-x, --expiration TEXT - setta il tempo in secondi prima che il contratto finisca, di default è 30 secondi

-f, --force-unique - forza la transazione ad essere unica, questo consumerà banda extra e rimuoverà qualsiasi protezione

contro errori multipli provenienti dalla stessa transazione

-s, --skip-sign - specifica se un wallet sbloccato può essere usato per firmare la transazione

-d, --dont-broadcast - non trasmette la transazione alla rete (la stampa al stdout)

-r, --ref-block TEXT - setta i riferimenti del numero del blocco o l'id del blocco usato per la TAPOS

-p, --permission TEXT - un account ed un livello di permesso, come in account@permission (default è account@active)

--max-cpu-usage-ms UINT - setta un tetto massimo di millisecondi per l'utilizzo della cpu, per l'esecuzione del

contratto (default è 0 cioè senza limiti)
--max-net-usage UINT - setta un tetto massimo di net da usare, in bytes, per la transazione (default è 0 cioè non ci sono limiti)
--add-code - aggiunge il livello di permesso eosio.code all'account active authority
--remoe-code - rimuove il livello di permesso eosio.code all'account active authority

Usage:

```
cleos set code someaccount1  
./path/to/wasm
```


- set abi

Crea o aggiorna l'abi di un account

Positionals:

account TEXT - l'account per settare il codice

abi-file TEXT - il percorso completo che contiene il contratto o WAST o WASM (richiesto)

Options:

-h, --help - stampa un messaggio di aiuto ed esce

-a, --abi TEXT - l'ABI per il contratto

-x, --expiration TEXT - setta il tempo in secondi prima che il contratto finisca, di

default è 30 secondi

-f, --force-unique - forza la transazione ad essere unica, questo consumerà banda extra e rimuoverà qualsiasi protezione contro errori multipli provenienti dalla stessa transazione

-s, --skip-sign - specifica se un wallet sbloccato può essere usato per firmare la transazione

-d, --dont-broadcast - non trasmette la transazione alla rete (la stampa al stdout)

-r, --ref-block TEXT - setta i riferimenti del numero del blocco o l'id del blocco usato per la TAPOS

-p, --permission TEXT - un account ed un livello di permesso, come in account@permission (default è

account@active)

--max-cpu-usage-ms UINT - setta un tetto massimo di millisecondi per l'utilizzo della cpu, per l'esecuzione del contratto (default è 0 cioè senza limiti)

--max-net-usage UINT - setta un tetto massimo di net da usare, in bytes, per la transazione (default è 0 cioè non ci sono limiti)

Usage:

```
cleos      set      abi      someaccount1  
./path/to/abi.abi
```

- transfer

Permette di trasferire un token

Positionals:

sender TEXT - l'account che deve mandare i token

recipient TEXT - l'account che deve ricevere i token

amount TEXT - il totale dei token da mandare e il simbolo del token

memo TEXT - il memo per il trasferimento

Options:

-c, --contract TEXT - il contratto che controllerà il token, di default è eosio.token


```
SYS", "memo": "hello world"}
```

- net connect

Connessione ad un peer.

Positionals:

host TEXT - l' hostname:port a dove collegarci

Options:

none

Usage:

esempio

cleos net connect
http://somehost.com:1234

- net disconnect

Disconnessione da un peer.

- net status

Mostra lo stato di un peer definito tramite l'url.

host TEXT - l' hostname:port a dove fare la richiesta dello stato

Options:

none

Usage:

esempio

```
cleos net status http://somepeer:1234  
#returns status of a configured peer
```

- net peers

Ritorna un file json del peer attualmente connesso.

Positionals:

none

Options:

none

Usage:

cleos net peers

#returns JSON of currently connected
peers

- push action

Spinge una transazione con una singola

azione.

Positionals:

contact TEXT - l'account che fornisce il contratto da eseguire

action TEXT - l'azione da eseguire sul contratto

data TEXT - gli argomenti del contratto

Options:

-h, --help - stampa un messaggio di aiuto ed esce

-x, --expiration TEXT - setta il tempo in secondi prima che il contratto finisca, di default è 30 secondi

-f, --force-unique - forza la transazione ad essere unica, questo consumerà banda

extra e rimuoverà qualsiasi protezione contro errori multipli provenienti dalla stessa transazione

-s, --skip-sign - specifica se un wallet sbloccato può essere usato per firmare la transazione

-j, --json - stampa il risultato come json

-d, --dont-broadcast - non trasmette la transazione alla rete (la stampa al stdout)

--return-packed - usato insieme a --dont-broadcast per prendere la transazione impacchettata

-r, --ref-block TEXT - setta i riferimenti del numero del blocco o l'id del blocco usato per la TAPOS

-p, --permission TEXT - un account ed un livello di permesso, come in

account@permission (default è account@active)

--max-cpu-usage-ms UINT - setta un tetto massimo di millisecondi per l'utilizzo della cpu, per l'esecuzione del contratto (default è 0 cioè senza limiti)

--max-net-usage UINT - setta un tetto massimo di net da usare, in bytes, per la transazione (default è 0 cioè non ci sono limiti)

--delay-sec UINT - setta i secondi delay_sec di default è 0

Usage:

esempio

cleos push action eosio.token transfer

```
'{"from":"bob","to":"alice","quantity":"2  
SYS","memo":"some SYS for you  
alice!"}' -p bob@active
```

- push transaction

Spinge una transazione arbitraria json

Positionals:

transaction TEXT - il file json da spingere o il nome del file che contiene la transazione

Options:

none

Usage:

esempio

```
cleos push transaction {}
```


Guida keosd

Il programma keosd si trova nella cartella eos/build/programs/keosd nelle repository di EOSIO/eos, può essere usato per salvare le chiavi private che cleos utilizza per firmare le transazioni da mandare sulla blockchain, keosd gira in locale e salva le chiavi private localmente.

Per la maggior parte degli utenti il modo migliore per usare keosd è quello di lanciarlo automaticamente, i file del wallet (per esempio foo.wallet) verranno creati in questa directory di default.

Anche se a keosd possiamo accedere dall'esterno tramite API, questo è stato concepito per essere operato localmente quindi si sconsiglia di usare per applicazioni wen.

Auto locking

Di default keosd è settato per auto bloccarsi dopo 15 minuti di inattività, questo è configurabile nel file config.ini, se volete disabilitare questa funzione bisogna settare un numero enorme e settare il valore a 0 causerà che keosd bloccherà sempre il wallet.

Lanciare keosd manualmente

E' possibile lanciare keosd manualmente semplicemente con il seguente comando:

```
$ keosd
```

Di default keosd crea una cartella `~/eosio-wallet` che inserisce il file `config.ini` dove questa cartella può essere specificata con gli argomenti `--config-dir`, mentre per specificare la cartella dei dati del wallet si può usare l'argomento `--data-dir`, il file contiene l' `http server endpoint` per le connessioni in entrata ed altri parametri per la condivisione delle risorse, da notare che se permettete a cleos di lanciare keosd

in automatico un file config.ini verrà generato che sarà diverso da quello lanciato in manuale.

Fermare keosd

Il modo migliore per fermare keosd è trovare il relativo processo e mandare un segnale SIGTERM allo stesso, o anche gli altri processi creati in automatico:

```
$ pkill keosd
```

Altre opzioni

Per una lista completa delle possibili opzioni possiamo avviare keosd --help:

```
$ keosd --help
```

Application Options:

Config Options for eosio::http_plugin:

```
        --http-server-address      arg  
(=127.0.0.1:8888)
```

The local IP and port to listen for incoming http connections; set blank to disable.

```
        --https-server-address arg      The  
local IP and port to listen for incoming https
```

connections; leave blank

to disable.

`--https-certificate-chain-file` arg

Filename with the certificate chain to present on https connections. PEM format.

Required for https.

`--https-private-key-file` arg

Filename with https private key in PEM format.

Required for https

`--access-control-allow-origin` arg

Specify the Access-Control-Allow-Origin to be returned on each request.

`--access-control-allow-headers` arg

Specify the Access-Control-Allow-Header

to be returned on each request.

`--access-control-max-age arg`
Specify the Access-Control-Max-Age to be returned on each request.

`--access-control-allow-credentials`
Specify if Access-Control-Allow-Credentials: true should be returned on each

request.
`--max-body-size arg (=1048576)`
The maximum body size in bytes allowed for incoming

RPC requests

`--verbose-http-errors`

Append the error log to HTTP responses

`--http-validate-host arg (=1)` If set

to false, then any incoming

"Host" header

is considered valid

`--http-alias arg`

Additionally acceptable values for the

"Host" header

of incoming HTTP

requests, can

be specified multiple

times. Includes

`http/s_server_address`

by default.

Config Options for `eosio::wallet_plugin`:

`--wallet-dir` arg (=".") The path of the wallet files (absolute path or relative to application data dir)

`--unlock-timeout` arg (=900) Timeout for unlocked wallet in seconds (default 900 (15 minutes)). Wallets will automatically lock after specified number of seconds of inactivity.

Activity is defined as any wallet command e.g. `list-wallets`.

`--yubihsm-url` URL

Override default URL of

http://localhost:12345 for connecting
to yubihsm-
connector

--yubihsm-authkey key_num

Enables YubiHSM support using given
Authkey

Application Config Options:

--plugin arg Plugin(s)

to enable, may be specified

multiple times

Application Command Line Options:

-h [--help] Print this

help message and exit.

-v [--version] Print

version information.

`--print-default-config` Print
default configuration template

`-d [--data-dir] arg`
Directory containing program runtime
data

`--config-dir arg` Directory
containing configuration
files such as
`config.ini`

`-c [--config] arg (=config.ini)`
Configuration file name relative to
`config-dir`

`-l [--logconf] arg (=logging.json)`
Logging configuration file name/path
for library
users

WIF - Wallet Import Format Specification

Il WIF (Wallet Import Format) è un sistema di codifica per chiavi private EDSA, EOS usa la stessa versione, checksum e schema di codifica degli indirizzi WIF di Bitcoin, quindi dovrebbe essere compatibile con le librerie esistenti, un esempio di chiave privata WIF:

5HpHagT65TZzG1PH3CSu63k8DbpvD8

Questa codifica è utile perché:

- copia ed incolla le chiavi private

(permettendo che l'intera chiave è copiata);

- include la chiave in un testo od essere modificata in un formato file:

- riduce la lunghezza della chiave.

Questa codifica non va bene perché:

- scrivere la chiave a mano dato che anche una sola modifica di una lettera può causare problemi;

- quando il codice della chiave è salvato sul computer ed i dati sono stati già controllati.

Alcune considerazioni:

Base58check

Base58check è un'implementazione JavaScript di questo algoritmo e può essere usato per codificare e decodificare le chiavi private EOS WIF:

```
base58check = require('base58check')
wif = base58check.encode(privateKey =
'00'.repeat(32), version = '80', encoding
= 'hex')
assert.equal('5HpHagT65TZzG1PH3CSu
wif)
let {prefix, data} =
base58check.decode(wif)
assert.equal(prefix.toString('hex'), '80')
```

```
assert.equal(data.toString('hex'),  
'00'.repeat(32))
```

Capitolo 5

Aspetti negativi di EOS

Critiche su EOS

Una critica che viene fatta nei confronti di EOS, riguarda la coordinazione negativa e per spiegarla dobbiamo partire dalla relativa teoria dei giochi riguardo la coordinazione.

	A	B
A	(2,2)	(0,0)
B	(0,0)	(4,4)

Nella precedente matrice vediamo che ci sono 2 punti di equilibrio di Nash (un profilo di strategia, per ogni giocatore, rispetto al quale nessun giocatore ha interesse ad essere l'unico a cambiare, ma è necessario che tutti agiscano insieme) che sono rappresentati da $[A,A]$ e $[B,B]$ e dove in questo caso dato che $[B,B]$ è più conveniente bisogna convincere gli altri ad andare da $[A,A]$ a $[B,B]$.

Infatti la fondamentale differenza tra il dilemma del prigioniero, dove entrambi i giocatori devono scegliere la soluzione $[B,B]$ perché è quella che garantisce un maggior payoff anche se $[A,A]$ ha una soluzione moralmente migliore, e il problema della

coordinazione è che non riguarda la moralità o il payoff, ma l'incentivo che una persona riceve per andare da una soluzione all'altra e quindi perché un gruppo numeroso di persone dovrebbe cambiare il modo di come fanno le cose attualmente?

Il sistema DPoS potrebbe usare la teoria dei giochi della coordinazione a proprio svantaggio e ci potrebbe essere uno scenario che i BP favoriscono e che non sono d'accordo con il resto dei nodi e

dove un masternode vuole il passaggio della catena da A a B questo sarebbe molto difficile e B potrebbe opporsi a questo passaggio, dove l'ecosistema EOS si basa su un sistema di votazione.

Quindi come vediamo l'ecosistema di EOS si basa molto sul meccanismo del voto, rilevando alcune problematiche:

- una partecipazione limitata;
- la tragedia dei beni comuni, si intende una situazione in cui diversi individui utilizzano un bene comune per interessi propri e quindi non è garantito il fatto che chi trarrà dei benefici dall'uso della risorsa sosterrà anche i costi, dove nella situazione attuale non ci sono utenti

molto attivi, oppure proxano il loro voto e quindi in modo passivo lasciano che siano altri a votare per lui, rispetto al totale degli account che ci sono:

EOS Block Producer Voting Statistics



Actual EOS Votes: 302,367,869.6395 EOS (29.68%)
Total Staked: 481,574,151.4100 EOS (47.28%)
Total EOS Supply: 1,018,644,037.9639 EOS
Number of voting accounts: 61,603 accounts
Total number of EOS accounts: 1,288,873 accounts

The stats on eosaauthority.com was last updated on 5th July 2019 13:01:02 UTC.

- gli interessi di chi holda il token e di chi utilizza il token non sono allineati e quindi si cerca di creare anche proposte mirate per incrementare il prezzo a scapito di chi utilizza il token per fare altro.

Sempre in tema di BP alcuni potrebbero affermare che con solo 21 BP e dove solo 15 di 21 rendono irreversibili le transazioni, allora questo sistema si presta bene alla centralizzazione ed in questo caso a soli 21 operatori, anche se questo per il momento è una possibilità, si sta cercando di mitigare al tutto, inoltre dobbiamo ricordare che anche le altre blockchain decentralizzate come Bitcoin ed Ethereum, che utilizzano il protocollo PoW, invece, hanno il problema dell'accentramento dell'hash power dove, anche grazie alle mining pool, ci potrebbe essere la centralizzazione del relativo potere e quindi in teoria compromettere l'intero

sistema nel caso questi si dovessero coalizzare in un modo o nell'altro, quindi il problema della centralizzazione, purtroppo, coinvolge quasi tutte le blockchain ma che ognuna cerca di risolvere come meglio crede.

Un'altra critica è quella che possiamo fare per quanto riguarda il meccanismo delle transazioni a costo 0, perché come detto in precedenza servono degli EOS per poter operare sulla blockchain di EOS e quindi questo implica da una parte che le persone ad investire per comprare dei token e poi successivamente metterli in stake per poter operare, comportando anche un investimento non indifferente quando in futuro il costo per singola unità potrebbe

raggiungere le 4 cifre decimali, e dall'altra che si taglia fuori tutti coloro che sono utenti occasionali e quindi non interessati ad un utilizzo del lungo periodo, cosa che non è vista positivamente da molte persone.

Infine c'è anche il problema niente in stake, ossia un validatore potrebbe mettere in stake per minare parallelamente delle catene di blocchi ed in teoria anche forzare una specie di hardfork, cosa che in un sistema PoW (Proof of Work) comporterebbe un dispendio di risorse non indifferente mentre in un sistema PoS (Proof of Stake), questo non accade dato che di base non ci sono ripercussioni sulla cosa, ma alcune blockchain punisco

severamente questo comportamento come fa Casper con la blockchain di Ethereum, e dove nella blockchain di EOS le punizioni non sono sufficienti dove il validatore scorretto ha solo una perdita di reputazione, cosa che non è sufficiente come sistema.

Altro aspetto negativo riguarda la RAM, dato che sempre più sviluppatori creeranno dApp e sempre più dati saranno salvati per un lungo periodo e quindi sempre più RAM sarà richiesta facendo diventare il costo della stessa eccessivo anche se ci sono alcune soluzioni per questo inconveniente:

- incrementando la RAM totale, infatti la blockchain di EOS è configurata con un

totale di 64 GB di RAM e dove B1 ha rilasciato un update del contratto di sistema di EOSIO dove permette ai BP di specificare il tasso di incremento della RAM;

- ridurre l'utilizzo della memoria degli account, B1 fornirà un aggiornamento al sistema di contratto per ridurre il minimo di RAM richiesta per creare un account di circa il 50%;

- un sistema di account gratuiti tramite il wallet iOS di B1, infatti si sta costruendo un hardware wallet gratuito utilizzando il sistema di Apple per offrire account gratuiti.

Inoltre sempre a causa della RAM e

che questa potrebbe aumentare in maniera esponenziale in molti ne comprano più del dovuto e quindi non la utilizzano riducendo di fatto la quantità disponibile per gli altri, e dove utilizzando l'algoritmo Bancor, che tra l'altro è stato settato con un parametro di 0.05% invece che di 50%, aggiustando questo parametro si avrà una migliore efficienza del mercato, fornendo un più accurato e prevedibile andamento del mercato.

Altra soluzione a questo problema sarebbe quello di bruciare i token che provengono dalla RAM e quindi i token EOS acquisterebbero più valore rispetto alla RAM e di conseguenza il prezzo della stessa diminuirà in favore di

quello del token.

Indice

Capitolo 1 - Introduzione

- Cos'è EOS
- Un po' di storia di EOS
- caratteristiche della blockchain di EOS

Capitolo 2 - Come iniziare

- Come creare account EOS
- Creare account tramite Cleos
- Account EOS Premium
- Piazzare offerte per account premium

con Cleos

- Account con multi firma
- Come si crea un account multi firma
- Settare permessi tramite Cleos
- Come scegliere il wallet EOS
- Creare e gestire un wallet con Cleos
- Come ottenere EOS
- Come vedere il prezzo di EOS
- Come mandare e ricevere EOS

Capitolo 3 - Come funziona la blockchain di EOS

- Come vengono confermate le transazioni in un sistema DPoS
- Una funzionalità di EOS, la TAPOS
- Le transazioni in EOS
- Alcuni tipi di transazioni su un account EOS
 - Transazione - New Account (newaccount)
 - Transazione - Buy RAM (buyram)
 - Transazione - Buy RAM Bytes (buyrambytes)
 - Transazione - Delegate Bandwidth

(delegatebw)

- Transazioni - Undelegate Bandwidth

(undelegatebw)

- Transazione - Vote Producer

(voteproducer)

- Transazione - Claim Rewards

(claimrewards)

- Transazione - Update Authentication

(updateauth)

- Transazione - Transfer (transfer)

- Transazione - Airdrop Action(airdrop)

- Come il wallet interagisce con la blockchain di EOS

- Cosa sono la RAM, CPU e NET

- RAM

- Come comprare e vendere RAM

- Come comprare e vendere RAM con Cleos

- CPU
 - NET
 - Come mettere in stake CPU e NET con Cleos
 - Come togliere dallo stake CPU e NET con Cleos
 - Come vedere il totale in stake con Cleos
 - I Block Producer (BP)
 - Ruolo del Block Producer
 - Come si vota per un BP
 - Come si diventa proxy con Cleos
 - Come si vota per un BP con Cleos
 - Funzione whitelist e blacklist dei BP
-
- REX (Resource Exchange)

- Come funziona REX
- Requisiti per utilizzare REX
- Prezzo di REX
- Come calcolare i prestiti di REX
- Inizializzare la REX pool
- La risoluzione del prestito
- Limite minimo saldo unlent
- Aggiustare il saldo della REX pool virtuale
- Derivate delle equazioni
- Utilizzare REX con un'interfaccia web
- Utilizzare REX con cleos
- Costituzione di EOS
- Gli Smart Contract
- Smart Contract nella blockchain di EOS
- Installare i CDT
- Homebrew (MacOS X)

- Ubuntu (Debian)
- CenOS/Redhat (RPM)
- Installare dai sorgenti
- Hello World
- Sicurezza negli smart contract
- Comprendere i file ABI
- Creare un file ABI

Capitolo 4 - nodeos, cleos e keosd

- Guida nodeos
- Avviare nodeos
- Configurazione nodeos
- Configurare il percorso
- Opzioni nodeos
- Settare nodeos
- Producing node
- Non-producing node
- Ambiente di sviluppo
- Local single-node testnet
- Local multi-node testnet
- Riprodurre nodeos
- Alcune problematiche
- Modalità lettura
- Logging

- Guida cleos
- Comandi di cleos
- Guida keosd
- Auto locking
- Lanciare keosd manualmente
- Fermare keosd
- Altre opzioni
- WIF - Wallet Import Format Specification

Capitolo 5 - Aspetti negativi di EOS

- Critiche su EOS

Note sull'autore

Alfredo A. de Candia è un piccolo sviluppatore Android indipendente con all'attivo una quindicina di app dove troviamo un database personale di virus, uno strumento per cercare diverse transazioni tra oltre 20 blockchain differenti, un dizionario di Kanji giapponesi, ed il database più completo sui compleanni dei personaggi di anime e manga con oltre 15000 voci, una must app per tutti gli otaku.

Studente della lingua giapponese da oltre 9 anni, questo gli ha permesso di apprendere una lingua orinetale e di

approfondirla sotto quell'aspetto portando alla creazione proprio di un dizionario dei Kanji giapponesi con un metodo innovativo per le app di quel genere, inoltre sempre nell'ambito dell'approfondimento della lingua giapponese e delle cultura del Giappone ha anche scalato il Monte Fuji, simbolo per eccellenza del Giappone reso famoso proprio dalle opere di Hiroshige, poco più che ventenne, acquisendo una conoscenza della cultura che va oltre il semplice turista passeggero.

La sua passione per la tecnologia e le nuove tecnologie applicate in diversi settori hanno permesso che si affacciasse a diverse realtà come quella

dello sviluppo Android e della tecnologia blockchain dove con il suo piccolo full node di bitcoin cercò di mettere in pratica quello che aveva appreso per poi spostarsi ad una blockchain nuova che era quella offerta da EOS diventandone da subito affasciato per la sua velocità e la quantità di dApp disponibili per quell'ecosistema e dove ha sempre cercato di imparare ad utilizzarle per comprenderne meglio il loro funzionamento.

Lo spirito divulgativo gli ha permesso di creare il suo canale YouTube dove con i suoi video tutorial e guide ha sempre cercato di informare e formare tutti gli appassionati e non del

mondo della blockchain mettendosi in gioco in prima persona anche con proposte proprio sulla blockchain di EOS, come la proposta di referendum relativa alla ricompensa dei votanti invogliando gli stessi a votare, ma anche tramite il supporto offerto ai vari eventi in materia blockchain nella sua zona dimostrando come a piccoli passi è possibile contribuire in prima persona a far accrescere il livello di consapevolezza delle persone che partecipano ai relativi eventi.

Infine è doveroso citare anche il suo apporto, come contributore, alla testa giornalistica, specializzata nel relativo settore della blockchain e non, permettendo di raggiungere anche coloro

che si trovano su tutto il territorio internazionale e non dato che gli articoli vengono anche pubblicati in lingua inglese e quindi permettere di entrare in comunicazione anche con persone di diversa nazionalità e culture differenti.

[1] <http://frankchester.com/chestahedron-geometry/>

[2] <https://onedrive.live.com/view.aspx?resid=FCC088B821D5FEB9!274&cid=fcc088O69Y>

[3] <https://eoskey.io/>

[4] <https://eos-account-creator.com/>

[5] <https://www.eosx.io/guides/how-to-create-account>

[6] <https://get-scatter.com/>

[7] <https://eostoolkit.io/multisig/create>

[8] <https://www.alohaeos.com/tools/benchmarks>

[9] <https://www.eoscharge.io/>

[10] <https://medium.com/@bytemaster/proposal-for-eos-resource-renting-rent-distribution-9afe8fb3883a>

[11]

<https://github.com/EOSIO/eosio.contracts/issue>

[12] [https://github.com/EOS-](https://github.com/EOS-Mainnet/governance/blob/master/eosio.system/regproducer-rc.md)

[Mainnet/governance/blob/master/eosio.system/
regproducer-rc.md](https://github.com/EOS-Mainnet/governance/blob/master/eosio.system/regproducer-rc.md)

[13] [https://github.com/EOS-](https://github.com/EOS-Mainnet/governance/blob/3ff0a9615b267bdf44)

[Mainnet/governance/blob/3ff0a9615b267bdf44](https://github.com/EOS-Mainnet/governance/blob/3ff0a9615b267bdf44)
[clause-constitution-rc.md](https://github.com/EOS-Mainnet/governance/blob/3ff0a9615b267bdf44)

[14] [https://eosauthority.com/polls_details?](https://eosauthority.com/polls_details?proposal=eosuseragree_20190207&lnc=en)
[proposal=eosuseragree_20190207&lnc=en](https://eosauthority.com/polls_details?proposal=eosuseragree_20190207&lnc=en)

[15] <https://www.npmjs.com/package/bs58>