WenQuanYi Micro Hei [Scale=0.9]WenQuanYi Micro Hei Mono song-WenQuanYi Micro Hei sfWenQuanYi Micro Hei "zh" = 0pt plus 1pt

nebulas Documentation

VersÃčo 1.0

nebulas

nov 21, 2019

Category:

CHAPTER 1

Welcome to the open-source Nebulas wiki!

Nebulas Ãl' uma blockchain pÃžblica de prÃşxima geraÃğÃčo, e visa criar um ecossistema de melhoria contÃ■nua. Com base no seu mecanismos de avaliaÃğÃčo, a Nebulas propÃţe incentivos orientados para o futuro, sistemas de consenso, e a habilidade de auto-evoluÃğÃčo sem biforcaÃğÃčo.

Este arquivo de documentos foi criado pela comunidade da Nebulas e qualquer um pode contribuir e construir um mundo descentralizado connosco. Convidamos toda a gente a ajudar.

A Wiki da Nebulas Ãl' uma ferramenta de colaboraÃğÃčo para a comunidade, para que esta seja capaz de publicar vÃąrios documentos. Estes incluem roteiros, livros brancos, guias de utilizaÃğÃčo da wiki, guias de desenvolvimento, recursos de aprendizagem, e outros documentos Ãžteis.

CHAPTER 2

Use Wiki

2.1 Mainnet

2.2 Dapps

2.3 Ecosystem

2.4 Get Involved

2.4.1 Guia de UtilizaÃğÃčo da Wiki

Como Editar uma PÃągina da Wiki

Um tutorial completo para modificar a Wiki pode ser encontrado aqui.

Software

Utilizadores com familiaridade com o git que prefiram editar a Wiki localmente deverÃčo utilizar reST para editar ficheiros .rst, e Pandoc Markdown para editar ficheiros .md.

Clique aqui para aprender a diferenÃğa entre Markdown e reST.

Eis alguns recursos de aprendizagem referentes ao Markdown:

- Como Utilizar Markdown por John Gruber
- Guia Markdown or iA Writer

ApÃşs Modificar

Quando modificar pÃąginas no Github, deve sempre clicar em "Preview changes" para visualizar as suas modificaÃğÃţes.

Depois da sua contribuiÃğÃčo ter sido aceite, pode verficar o processo de compilaÃğÃčo aqui.

2.4.2 Como Contribuir

A sua contribuiÃğÃčo Ãl' importante!

Nebulas visa criar um ecossistema de melhoria contÃ■nua, o que significa que precisamos da ajuda da comunidade. ÃL' aqui que pedimos que contribuam. NÃčo exclusivamente em termos de programaÃğÃčo, mas tambÃl'm relatÃşrios de bugs e traduÃğÃţes, e tambÃl'm a nÃ■vel de comunicaÃğÃčo com outros membros da comunidade e esclarecimento de dÃžvidas com os mais novos.

Uma grande parte dos nossos projectos pode ser encontrada aqui.

1. CÃşdigo & DocumentaÃğÃčo

1.1. Desenvolvimento da Mainnet

Para alÃl'm da programaÃğÃčo, o desenvolvimento da mainnet tambÃl'm precisa de enfrentar alguns dos problemas mais desafiantes do mundo da blockchain. Por exemplo, precisamos de engendrar mecanismos resistentes Ãă manipulaÃğÃčo para a blockchain, melhorar a seguranÃğa da mainnet, implementar novos algoritmos criptograficos, etcetera.

Estamos entusiasmados por podermo-nos dedicar na Ã■ntegra Ãă blockchain, e ver como ela melhora a vida de todos nÃşs. Como tal, queremos partilhar esta experiÃłncia excitante com a comunidade. Este Ãl' o nosso convite para os desenvolvedores!

Aprenda mais:

- O nosso github: https://github.com/nebulasio/go-nebulas
- O nosso roteiro: https://nebulas.io/roadmap.html (Stay tuned)

1.2. RelatÃşrios de Bugs

Sempre valorizÃąmos a submissÃčo de bugs!

Se encontrar um bug, por favor submeta um relatÃşrio Ãă comunidade Nebulas. IrÃą ser recompensado por isso! Visite o programa de recompensas da Nebulas aqui.

Bugs podem ser encontrados na testnet, mainnet, nebPay, neb.js, na web wallet, tal como em outras ferramentas e atÃl' documentaÃğÃčo. Seguiremos o sistema de avaliaÃğÃčo de risco OWASP para calcular a recompensa correspondente, baseada no grau de importÃćncia do bug.

Se tiver sugestÃţes para a resoluÃğÃčo de bugs, ou relativamente a melhorias, por favor nÃčo hesite em entrar em contacto connosco. Pode tambÃl'm participar no desenvolvimento directamente, e proteger activos na blockchain. Juntos, vamos tornar a Nebulas mais segura, resiliente, e robusta.

Para submeter relatÃşrios e informaÃğÃčo sobre os mesmos bugs, por favor poste no mail group da Nebulas relevante. Quando o fizer, tenha cuidado de modo a prevenir a exploraÃğÃčo dos bugs, tal como a submissÃčo de relatÃşrios duplicados. Convidamo-lo a seguir o mail group e a juntar-se Ãă discussÃčo.

Lista dos Mail groups: https://lists.nebulas.io/cgi-bin/mailman/listinfo

Lista dos bugs da Mainnet: https://lists.nebulas.io/cgi-bin/mailman/listinfo/mainnet-bugs

Lista dos bugs da Testnet: https://lists.nebulas.io/cgi-bin/mailman/listinfo/testnet-bugs

1.3. TraduÃğÃčo

A traduÃğÃčo Ãl' importante para espalhar Nebulas pelo mundo inteiro!

Todos os membros da comunidade Ãă volta do mundo sÃčo encorajados a participar na traduÃğÃčo de documentaÃğÃčo da Nebulas. Pode traduzir tudo da wiki, incluÃ■ndo documentos tÃl'cnicos de desenvolvimento da mainnet, FAQ DApp, documentos oficiais como o Livro Branco e o Livro Amarelo, a IntroduÃğÃčo aos PrincÃ■pios de Design da Nebulas, e mais. A sua contribuiÃğÃčo tem o potencial de ajudar uma imensidÃčo de desenvolvedores e membros da comunidade. Por favor note que alguns documentos vÃčo necessitar de um background acadÃl'mico em matemÃątica, ciÃłncia de computadores, criptografia, e/ou outras especialidades.

1.4. CriaÃğÃčo de DocumentaÃğÃčo

Desenvolvedores da comunidade da Nebulas precisam de documentaÃğÃčo para entender e usar as vÃąrias funÃğÃţes disponÃ■veis na Nebulas. A comunidade Ãl' encorajada a criar introduÃğÃţes tÃl'cnicas e FAQs. Adicionalmente, membros da comunidade tambÃl'm precisam de introduÃğÃţes de digestÃčo fÃącil, e guias de utilizaÃğÃčo das vÃąrias ferramentas do ecossistema.

A sua contribuiÃğÃčo serÃą benÃl'fica para todos os desenvolvedores e membros da comunidade, e poderÃą tambÃl'm ser traduzida para vÃąrias lÃ■nguas de modo a alcanÃğar um maior nÃžmero de utilizadores.

1.5. Design do UI da Wiki

Convidamos desenvolvedores de UI a optimizar a nossa pÃągina de wiki, e a tornÃą-la ainda mais legÃ■vel e fÃącil de usar.

FaÃğa download da nossa wiki > (design template)

FaÃğa download do nosso LOGO > Se tiver algum questÃčo ou comentÃąrio nÃčo hesite em postar no nosso github.

2. User Groups

ComunicaÃğÃčo Ãl' chave no que toca Ãă criaÃğÃčo de comunidades. As pessoas precisam de falar entre si e partilhar os seus pensamentos e ideias sobre a Nebulas.

A Nebulas utiliza vÃąrias plataformas para conectar a nossa comunidade global. Por favor refira-se ao link âĂIJCommunityâĂİ no website oficial para obter mais informaÃğÃčo: https://nebulas.io/community.html

Discord: DisponÃ■vel para todos os membros da comunidade. Pode tambÃl'm subscrever-se ao Nebulas News, e participar na discussÃčo de grupo. Discord Ãl' a primeira escolha de muitos utilizadores.

Mailing lists: Grupo de discussÃčo para desenvolvimento e relatÃşrios de bugs. Desenvolvedores sÃčo livres de se subscreverem.

Forum: Reddit/r/nebulas (para todos), Reddit/r/nasdev (para desenvolvedores)

ComunicaÃğÃčo: Slack (para desenvolvedores), Telegram (para utilizadores nÃčo-desenvolvedores)

Membros da comunidade sÃčo livres de criar um IRC (Internet Relay Chat) para melhor comunicaÃğÃčo entre desenvolvedores.

3. Recompensas

NÃşs, a equipa da Nebulas, introduzimos algumas recompensas para premiar contribuidores iniciais. Como foi mencionado acima, pode ver os projectos e recompensas disponÃ■veis aqui.

4. DoaÃğÃţes

Apreciamos doaÃğÃţes da comunidade para o desenvolvimento da Nebulas. Ambos NAS e ETH sÃčo aceites. TambÃl'm agradecemos ajuda em termos materiais, isto Ãl', a cedÃłncia de espaÃğo, guias locais, fotografia, etcetera. Caso queira, terÃą crÃl'dito publicamente pela sua contribuiÃğÃčo. Caso seja um membro com entusiasmo da nossa comunidade e queira contribuir, por favor contacte contact@nebulas.io para obter mais detalhes.

2.4.3 Programa de Recompensas

Practicamente todos os projectos sÃčo postados na PÃągina de Projectos da Nebulas juntamente com as suas recompensas correspondentes, e os utilizadores tÃłm a obrigaÃğÃčo de se candidatar de medida a candidatarem-se a um projecto ou partes dele. Este processo aplica-se aos Programas de Recompensas da Wiki e do NAT. Para jÃą o Programa de Recompensa de Bugs da Nebulas apenas requere que preencha um inquÃl'rito sobre o bug em questÃčo.

Abaixo encontrarÃą informaÃğÃčo detalhada sobre todos os Programas de Recompensa da Nebulas para que possa comeÃğar a contribuir para a prosperidade do ecossÃ■stema da Nebulas!

Programa de Recompensa da Wiki da Nebulas

PrÃl'viamente utilizadores que tomassem parte na criaÃğÃčo ou modificaÃğÃčo de conteÃždo da Wiki da Nebulas tinham direito a potencialmente receberem recompensas na forma de NAS. Hoje em dia, o processo Ãl' bastante diferente.

Para se qualificar para receber recompensas, vÃą ao pÃągina do projecto acima mencionada e use o motor de busca para procurar por "wiki", ou simplesmente clique aqui para ver todos os projectos relacionados com a wiki que estÃčo disponÃ■veis.

Programa de Recompensa de Bugs da Nebulas

O programa de Recompensa de Bugs da Nebulas visa melhorar a seguranÃğa do ecossÃ■stema de Nebulas, assegurando a estabilidade do mesmo. O programa de Recompensa de Bugs recompensa vulnerabilidades descobertas. Este programa de recompensas foi criado e implementado pelo ComitÃl' TÃl'cnico da Nebulas (NTC), em conjunto com a equipa tÃl'cnica da Nebulas, e com a comunidade. NTC encoraja a comunidade a fornecer informaÃğÃčo sobre vulnerabilidades de seguranÃğa atravÃl's do processo descrito abaixo, e a ter uma participaÃğÃčo activa na criaÃğÃčo do ecossÃ■stema da Nebulas, e ao mesmo tempo ser recompensado.

Categorias de Bugs

O Programa de Recompensa de Bugs divide as recompensas em 2 categorias, recompensa de bugs comuns e de bugs especiais. Os bugs comuns incluem vulnerabilidades descobertas na mainnet da Nebulas, na testnet, no nebPay, na Web wallet, no web.js e outros, enquanto que os bugs especiais incluem vulnerabilidades descobertas na funÃğÃčo de invocaÃğÃčo intercontractuais e outras.

Elegibilidade

O ComitÃl' TÃl'cnico da Nebulas irÃą avaliar o tamanho das recompensas de acordo com a gravidade do bug, de acordo com o MÃl'todo de AvaliaÃğÃčo de Risco OWASP baseado no **Impacto** e **Probabilidade**. No entanto, as recompensas sÃčo determinados e estÃčo exclusivamente ao critÃl'rio do comitÃl'.

Figura 1

Impacto:

- Alto: Bugs que afectam a seguranÃğa de activos.
- MÃl'dio: Bugs que afectam a estabilidade do sistema.

- Baixo: Outros bugs que nem afeitam a seguranÃğa de activos, nem do sistema.

Probabilidade:

- Alta: O bug pode ser descoberto por qualquer um que efectue uma dada operaÃğÃčo, independentemente do bug ter sido descoberto.
- MÃl'dia: Apenas um grupo selecto consegue descobri-los (tal como um bug que apenas pode ser encontrado por desenvolvedores, e utilizadores ditos normais nÃčo sÃčo afectados.)
- Baixa: Cobre menos de 1% de uma populaÃğÃčo especÃ■fica, como por exemplo problemas em modelos de Android raros, ou outros casos excepcionais.

Quantidade:

De modo a assegurar que a recompensa do denunciador do bug seja o esperado, a quantidade em dolares Americanos serÃą atribuÃ■da em NAS. A quantidade da recompensa estÃą dividida em 5 categorias:

- Critica: US$1,000 ou mais (Sem limite mÃąximo)
- High: US$500 ou mais
- Medium: US$250 ou mais
- Low: US$100 ou mais
- Improvement: US$30 ou mais

Nota: A recompensa especial da testnet da Nebulas (como uma relacionada com a chamada da funÃğÃčo inter-contracts) foi aumentada de acordo com as nossas prioridades, e a quantidade de dolares equivalentes serÃčo atribuÃ■dos em NAS.

Denuncie um Bug

Por favor faÃğa o seu relato aqui, atravÃl's deste link.

Notas:

1. Por favor assegure-se do rigor e claridade do conteÃždo, porque a avaliaÃğÃčo da recompensa serÃą baseada no conteÃždo submetido aqui.
2. Se muita gente descobrir o mesmo bug, a ordem cronolÃşgica irÃą ser usada para determinar o vencedor da recompensa. Utilizadores da comunidade sÃčo bem vindos a discutir os bugs, mas a discussÃčo em si sÃş nÃčo Ãl' considerada um relato, logo um relato tem de ser submetido.

Notas adicionais:

1. O programa Bug Bounty da Nebulas Ãl' de duraÃğao longa. O ComitÃl' Tecnico da Nebulas reserva o direito final Ãă interpretaÃğÃčo deste programa, e ao direito de ajus-

tar, ou cancelar as recompensas, eligibilidade, e quantidade.

2. O ComitÃl' Tecnico da Nebulas irÃą confirmar e avaliar o relatÃşrio do bug apÃşs a sua submissÃčo. O tempo da avaliaÃğÃčo irÃą depender da gravidade do problema e da dificuldade da sua resoluÃğÃčo. A avaliaÃğÃčo do mesmo serÃą enviada para o autor do relatÃşrio por email o mais rÃąpido possÃ■vel.
3. De modo a evitar a exploraÃğÃčo de bugs, os relatÃşrios devem ser submetidos pelo portal.
4. Os autores dos relatÃşrios irÃčo manter os bugs nÃčo-pÃžblicos e confidenciais atÃl' 30 dias apÃşs a submissÃčo do relatÃşrio Ãă Nebulas, e nÃčo irÃčo divugÃą-los a terceiros. O tempo de confidencialidade poderÃą ser extendido pela Nebulas unilateralmente. Caso este acordo seja quebrado, o autor do relatÃşrio serÃą responsÃąvel por todas as perdas e danos Ãă Nebulas e seus utilizadores, tal como a sua compensaÃğÃčo.
5. O ComitÃl' Tecnico da Nebulas encoraja membros da comunidade a conversar com a Equipa Tecnica da Nebulas e outros membros da comunidade nos grupos de discussÃčo pÃžblicos. TambÃl'm aceitamos ajuda na resoluÃğÃčo dos bugs. Junte-se Ãă maillist da Nebulas.

Programa de Recompensa de Bugs no NAT da Nebulas

O NAT inclui cerca de 7 smart contracts.

Para bugs relacionados com os smart contracts do NAT, pode ir aqui para reivindicar a sua recompensa. Note que tambÃl'm terÃą que preencher o inquÃl'rito a detalhar o seu bug, depois de se candidatar na pÃągina de projectos da Nebulas, para se tornar elegÃ■vel.

Os smart contracts podem ser actualizados a qualquer momento, e encontram-se disponÃ■veis nos seguintes links:

multisig: n1orrpFGmcQSvGrbKTD7RHweTPe61ut7svw

NAT NRC20: n1mpgNi6KKdSzr7i5Ma7JsG5yPY9knf9He7

distribute: n1uBbtFZK3Acs2T6JUMv6bSAvS6U6nnur6j

pledge_proxy: n1obU14f6Cp4Wv7zANVbtmXKNkpKCqQDgDM

pledge: n1zmbyLPCt2i8biKm1tNRwgAW3mhyKUtEpW

vote: n1pADU7jnrvpPzcWusGkaizZoWgUywMRGMY

NR_DATA: n21KaJxgFw7gTHR9A5VFYHsQrWdL61dCqvK

2.4.4 O que Ãl' a Nebulas

Nebulas: Blockchain PÃžblica de PrÃşxima GeraÃğÃčo

Nebulas visa construir um ecossistema de melhoria contÃ■nua.

Nebulas Ãl' uma blockchain pÃžblica de prÃşxima geraÃğÃčo. Introduz o Nebulas Rank (NR), uma nova medida de valor para qualquer unidade do universo da blockchain, como endereÃğos, DApps, e smart contracts. Baseado no NR, temos o Nebulas Incentive (NI), que motiva os desenvolvedores com o Developer Incentive Protocol, e os utilizadores com o algoritmo de consenso Proof of Devotion. Adicionalmente, propÃţe o Nebulas Force (NF), o que atribui Ãă blockchain e respectivos smart contracts uma capacidade auto-evolutiva. Em unÃ■sono, o NR, o NI, e o NF, produzem um ecossistema na blockchain em melhoria continua e expansÃčo, usando os princÃ■pios incluÃ■dos no artigo Nebulas Governance como guias da sua evoluÃğÃčo.

Existem trÃłs caracterÃ■sticas tÃl'cnicas: classificaÃğÃčo de valor, auto-evoluÃğÃčo, e o incentivo nativo.

Encarando os desafios e oportunidades acima descritos, visamos criar um sistema blockchain auto-evolutivo baseado em incentivos de valor.

PrincÃ■pios

A blockchain da Nebulas tem trÃłs princÃ■pios principais:

Nebulas Rank (NR)

O Nebulas Rank (NR) Ãl' um algoritmo de classificaÃğÃčo usado para medir a influÃłncia das relaÃğÃţes entre endereÃğos, smart contracts, e aplicaÃğÃţes distribuÃ■das (DApps). Ajuda utilizadores a utilizar informaÃğÃčo entre a quantidade sempre crescente de dados na blockchain, e tambÃl'm permite que desenvolvedores usem o nosso motor de busca directamente nas suas aplicaÃğÃţes. It helps both users utilize information among the ever-increasing amount of data on all blockchains & developers to use our search framework directly in their own applications.

Na Nebulas, medimos o valor relativo a:

- **Liquidez**

FinanÃğa Ãl' essencialmente as actividade sociais que optimizam recursos sociais, atravÃl's de liquidez de capital, e promovem desenvolvimento econÃşmico. As blockchains estabelecem uma rede de valor na qual activos financeiros podem fluir. O volume diÃąrio da Bitcoin e do Ethereum, as mais conhecidas de todos nÃşs, jÃą excede $1 biliÃčo. A partir destes dados podemos ver que quanto maior o volume e escala de transaÃğÃţes, maior a liquidez. Por sua vez, liquidez mais alta irÃą aumentar a quantidade de transaÃğÃţes e realÃğar o valor. Isso irÃą fortalecer cada vez mais o valor dos activos financeiros, criando um mecanismo de feedback positivo. Logo, liquidez, ex. frequÃłncia e escala das transaÃğÃţes, Ãl' a primeira dimensÃčo medida pelo NR.

- **PropagaÃğÃčo**

Plataformas sociais como o WeChat e o Facebook tÃłm quase cerca de 3 biliÃţes de utilizadores activos por mÃłs. O crescimento rÃąpido das plataformas sociais Ãl' o resultado

da reflexÃčo de plataformas sociais existentes e crescimento viral mais forte. Em particular, transmissÃčo viral, ex. velocidade, alcance, profundidade da transmissÃčo de informaÃğÃčo, e ligaÃğÃţes., Ãl' o Ãndice chave para monitorar a qualidade e crescimento das redes sociais. No mundo da blockchain, podemos observar o mesmo padrÃčo. PropagaÃğÃčo viral poderosa indica alcanÃğa e profundidade de liquidez de activos, o que pode promover a qualidade e escala dos activos do mundo da blockchain. Logo, transmissÃčo viral, ex. alcanÃğa e profundidade de liquidez dos activos, Ãl' a segunda dimensÃčo medida pelo NR.

- **Interopabilidade**

Durante o perÃodo inicial da internet, apenas haviam websites bÃąsicos e informaÃğÃčo privada. Agora, informaÃğÃčo em plataformas diferentes pode ser encaminhada na rede, e silos isolados de dados estÃčo a ser gradualmente desmantelados. Esta moda Ãl' o processo de identificar informaÃğÃčo de maior dimensÃčo. Do nosso ponto de vista, o mundo da blockchain deverÃą seguir um padrÃčo semelhante, mas a sua velocidade serÃą mais alta. A informaÃğÃčo sobre os activos dos utilizadores, smart contracts, e DApps, tornar-se-Ãă mais rica, e a interacÃğÃčo de informaÃğÃčo de dimensÃţes superiores serÃą mais fequente, logo melhor interopabilidade irÃą-se tornar mais importante. Logo, a terceira dimensÃčo medida pelo NR Ãl' a interopabilidade.

Baseado nas dimensÃţes acima referidas, construÃmos o sistema de NR da Nebulas, atravÃl's de dados mais ricos, construÃndo melhores modelos, descobrindo dimensÃţes de valor mais diversificadas, e estabelecendo uma medida de valor no mundo da blockchain.

Nebulas Force (NF)

Uma sÃl'rie de protocolos bÃąsicos como o NR, o PoD, o DIP, tornar-se-Ãčo parte dos dados da blockchain. Com o crescimento dos dados da Nebulas, estes protocolos bÃąsicos irÃčo ser actualizados, o que irÃą prevenir a fractura entre desenvolvedores e a comunidade, tal como a "bifurcaÃğÃčo" ("fork"). Chamamos a esta capacidade fundamental da blockchain de âĂIJNebulas ForceâĂİ (NF).

ÃĂ medida que a comunidade da Nebulas cresce, a habilidade do NF e de outros protocolos bÃąsicos em serem actualizados serÃčo abertos Ãă comunidade. De acordo com o pesos do NR de cada utilizador e o mecanismo de votaÃğÃčo da comunidade, quem determina a direcÃğÃčo da evoluÃğÃčo da Nebulas e os objectivos da sua actualizaÃğÃčo sÃčo os utilizadores. Com a ajuda da tecnologia central do NF e a sua abertura, Nebulas terÃą um potencial evolutivo sempre crescente, com um nÃžmero de possibilidades evolutivas infinÃto.

Nebulas Incentive (NI)

O Nebulas Incentive inclui o Proof of Devotion (PoD) e o Developer Incentive Protocol (DIP).

Proof-of-Devotion (PoD)

Baseado no sistema NR da Nebulas, iremos adoptar o algoritmo de consenso PoD (Proof-of-Devotion). O PoD dÃą a oportunidade a um utilizador "influencial" da Blockchain da Nebulas de se tornar um escriturÃąrio ("bookkeeper") e receber recompensas dos blocos da Nebulas e bÃşnus de transacÃğÃţes, o que o irÃą encorajar a contribuir para a estabilidade e seguranÃğa da Nebulas continuamente.

Developer Incentive Protocol (DIP)

Na Nebulas, propusemos o conceito do DIP (Developer Incentive Protocol) para desenvolvedores de smart contracts e DApps. O conceito central do DIP: no intervalo pre-especificado de blocos, os desenvolvedores cujos smart contracts e DApps tiverem sido implementados online no intervalo mais recente com um valor NR mais alto do que o limite especificado serÃčo recompensados pelo DIP. O valor correspondente serÃą em (NAS token), e esses incentivos serÃčo gravados nos blocos pelos bookkeepers. Com o mecanismo positivo de incentivo, mais e mais desenvolvedores irÃčo receber estÃ■mulos para criarem smart contracts e DApps valiosos, o que ajudarÃą a criar um ecossistema de feedback positivo na comunidade dos desenvolvedores.

Nebulas Community Governance

Orange Paper

O Livro Laranja da Nebulas foi lanÃğado a 30 de Abril de 2019 e destaca a forma tecnolÃşgica Ãžnica e inovativa como a Nebulas pode gerir activos pÃžblicos na chain de maneira a atingir os seus objectivos e implementar a sua visÃčo: âĂIJExplorar um novo modelo descentralizado de colaboraÃğÃčo, implementar uma organizaÃğÃčo autÃşnomo descentralizada que fornece incentivos positivos e tem a capacidade de auto-evoluÃğÃčo (Decentralized Autonomous Organization, DAO).âĂİ

Os princÃ■pios sobre os quais a GovernaÃğÃčo da Nebulas foi construÃ■da sÃčo os seguintes:

1. Mecanismo de supervisÃčo e estructura organizacional: Grupos da Comunidade da Nebulas irÃčo operar independentemente mas vÃčo estar constrangidos/limitados um pelo outro: o Nebulas Council, a Nebulas Foundation, o Nebulas Technical CommitteeâĂŁâĂŤâĂŁarticulando a sua composiÃğÃčo bÃąsica, poderes, e deveres;
2. ColaboraÃğÃčo on-chain: IntroduÃğÃčo do projecto comunitÃąrio da Nebulas, âĂIJNAT VotaÃğÃčo On-chainâĂİ conseguirÃą atingir a colaboraÃğÃčo da comunidade no processo de actualizaÃğÃčo dos sistemas.
3. Economias e incentivos: O design da economia do sistema de votaÃğÃčo e a forma da economia dar incentivos positivos a cada membro da comunidade durante o governo da Nebulas.

Aprenda mais sobre a GovernaÃğÃčo da Nebulas ao ler o nosso Livro Laranja aqui.

The NAT Token

NAT Ãl' o token NRC20 da Nebulas, derivado do Nebulas Rank, e Ãl' parte integral do sÃ■stema de voto da blockchain. A oferta total estÃą limitada a 100 biliÃţes.

Inicialmente, o sistema de voto 0 NAS foi utilizado. Um endereÃğo Ãl' criado por opÃğÃčo de voto, e o utilizador transfere 0 NAS para o endereÃğo referente Ãă opÃğÃčo escolhida. Para contar os votos das referenda onde cada pessoa apenas tem direito a um voto, as transferÃłncias sÃčo analizadas e endereÃğos duplicados sÃčo eliminados, e o nÃžmero de transacÃğÃţes sÃčo contadas. ÃL' um sistema bastante limitado.

O sistema de votaÃğÃčo de NAT tem vÃąrias formas de combater as fraquezas do sistema de voto 0 NAS.

- Introduz a possibilidade de "Peso dos Votos" para tipos especiais de eleiÃğÃţes.
- Cria incentivos para votar em forma de recompensas NAT.
- O poder de voto individual Ãl' determinado pelo Nebulas Rank e o seu involvimento com a comunidade e ecossistema da Nebulas.
- ProtecÃğÃčo contra fraudes devido Ãă oferta total em comparaÃğÃčo com NAS.

ObservaÃğÃčo: NAT transferido serÃą queimado.

Para informaÃğÃčo mais detalhada, leia o Livro Laranja da GovernaÃğÃčo da Nebulas.

Como Obter NAT

Existem vÃąrias maneiras para obter NAT.

- ***Via airdrops:*** Airdrops NAT ocorrem uma vez por semana e sÃčo baseados no Nebulas Rank de um endereÃğo. Para calcular a quantidade de tokens que pode receber atravÃl's dos airdrops, considere ler este artigo.
- ***Via pledging:*** pledging NAS permite que um individuo receba NAT tokens pela quantidade de tempo que o NAS esteja bloqueado num endereÃğo. Cancelar o seu pledge farÃą com que vocÃł volte a receber as suas NAS, mas irÃą parar de receber NAT.
- ***Via VotaÃğÃčo ou Nebulas Rank:*** se tiver um Nebulas Rank diferente de zero, estarÃą elegÃ■vel para receber incentivos de voto. De momento, o Ã■ndice do incentivo Ãl' 10. Como tal, o nÃžmero de if you have a non-zero Nebulas Rank you are elligible to receive voting incentives. Currently, the incentive index is set at 10. Thus, the number of voting incentives you will receive is equal to 10 times your Nebulas Rank that week, or 10 times the number of NAT that was sent out of that address for the week, whichever is lower.

Aprenda Mais

Onde Votar

Go.Nebulas.io Proposals.

Active nebulas.io Ballots.

Links Uteis (em InglÃłs)

nebulas.io NAT‘s Main Page.

Why NAT is a fundamental component of the Nebulas ecosystem.

NAS on-chain voting starts!

Three minutes to take you into the world of NAT.

Participate in Nebulas ecosystem voting & receive NAT incentives!

How to obtain NATâĂŁâĂŤâĂŁPart 1: How to Pledge your NAS.

How to obtain NATâĂŁâĂŤâĂŁPart 2: How to Pledge your NAS via offline mode.

How to obtain NATâĂŁâĂŤâĂŁPart 3: Receiving NAT incentives & how to improve your NR.

How to receive NAT without a Nebulas Rank.

ClassificaÃğÃčo de valor

De modo a possibilitar a descoberta de valor na blockchain, o **Nebulas Rank** analisa dados multidimensionais no mundo da blockchain e torna possÃ■vel a estructura do motor de busca descentralizado.

Auto-evoluÃğÃčo

Para prevenir danos causados pela bifurcaÃğÃčo da blockchain, o **Nebulas Force** permite actualizaÃğÃţes e iteraÃğÃţes rÃąpidas da sua blockchain sem necessitar de hard forks.

Incentivos nativos

Com mecanismos de incentivo e consenso voltados para o futuro, o **Nebulas Incentive** recompensa desenvolvedores e utilizadores que contribuam para a sustentabilidade e crescimento do ecossistema.

Isto Ãl’ um excerto do Livro Branco NÃčo-tÃl’cnico da Nebulas.

Se quiser saber mais sobre Nebulas, por favor subscreva-se ao blog oficial ou visite o nosso website: nebulas.io. Leia Livro Branco NÃčo-tÃl’cnico (PortuguÃłs), Livro Branco TÃl’cnico (English).

2.4.5 Go-Nebulas

Nebulas Technical Committee

s Nova Tech Tradeoffs(11.21.2018)

Summary

1. The process to submit IR (LLVM Intermediate Representation) and who can submit IR (LLVM Intermediate Representation)
2. The time window for NR & DIP
3. How much NAS for DIP & how to distribute NAS for DIP

Detailed minutesïijŽ

1. The process to submit IR and who can submit IR

1. Nebulas Nova will use an auth_table to decide whose IR can be executed and the lifetime of each IR.
2. auth_table is a set of tuples, and each tuple includes IR name, submitterâĂŹs address, the valid start and end height for the submitter.
3. Only the auth_adminâĂŹs auth_table can update in Nebulas Nova. The auth_admin account should be created by a cold wallet. Each IR should be managed by different accounts. Nebulas Technical Committee will further discuss the community governance details with the community. Before the we finalized the governance details , the Nebulas team will not recklessly open the IR submission access.The NBRE only executes several predefined IRs, like checking the auth_table, and the IRs defined in auth_table. Other IRs will not be executed
4. However, each node may change the code. And that could be the auth_admin account, and the auth_table. Consequently, it may change the behaviors in NBRE, and the node shall fail to sync data with the main-net

2. The time window for NR &DIP

1. In the yellow paper introducing Nebulas Rank, we have mentioned that to avoid the affect of loop attack, we will remove the forwarding loop before we calculate the In-and-Out degree for the transaction graph, thus the time-window is important for anti-manipulation.
2. If the time-window is too short, there may be more cheating.
3. For now, we suggest the time window in several days.
4. We should monitor the cheating status, and adjust the time-window if necessary.

5. time window for DIP should be much more larger than the time window of NR, for now, we suggests 7 days

3. How much NAS for DIP & how to distribute NAS for DIP

1. For each month, we suggest around 500 NAS in total for now, and adjust the amount subject to the DIP feedback in the future, the winners shall be relatively stable, so a winner will get reward in several months.
2. We will have a special account for distributing NAS for DIP. The account can only send special transactions for DIP.

About Nebulas Technical Committee

The Nebulas Technical Committee adheres to the spirit of openness, sharing, and transparency, and is committed to promoting the decentralization, and the community of the research and development of the Nebulas technology. Blockchain technology opens up possibilities for building new and self-motivated open source communities. NebulasâĂŹ technical concepts unclude mechanisms for evaluation, self-evolution, and self-incentives, which provide a guarantee for building a world of decentralized collaboration. The Nebulas Technical Committee will fully promote the realization of the Nebulas vision.

Subscribe to nebulas mailing list and learn the latest progress of Nebulas: mailing list

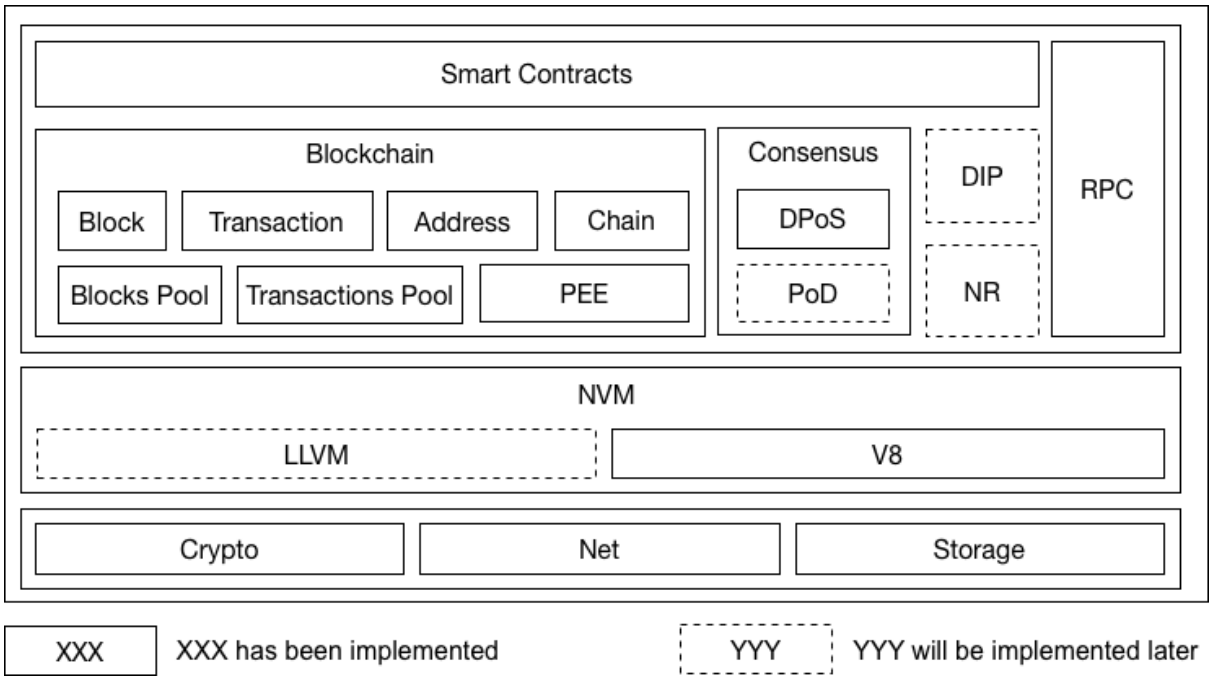
For more info, please visit Nebulas official website.

Papers

- O Livro Branco TÃl'cnico da Nebulas.
- O Livro Branco NÃčo TÃl'cnico da Nebulas.
- O Livro Amarelo da Nebulas âĂŤ o Nebulas Rank. Pode aceder ao repositÃşrio aqui.
- O Livro Malva da Nebulas âĂŤ o Developer Incentive Protocol. Pode aceder ao repositÃşrio aqui.
- O Livro Laranja da Nebulas âĂŤ Nebulas Governance. Pode aceder ao repositÃşrio aqui.

Como sempre, traduÃğÃţes e relatÃşrios de bugs sÃčo sempre bem vindos. Aprenda mais sobre como contribuir.

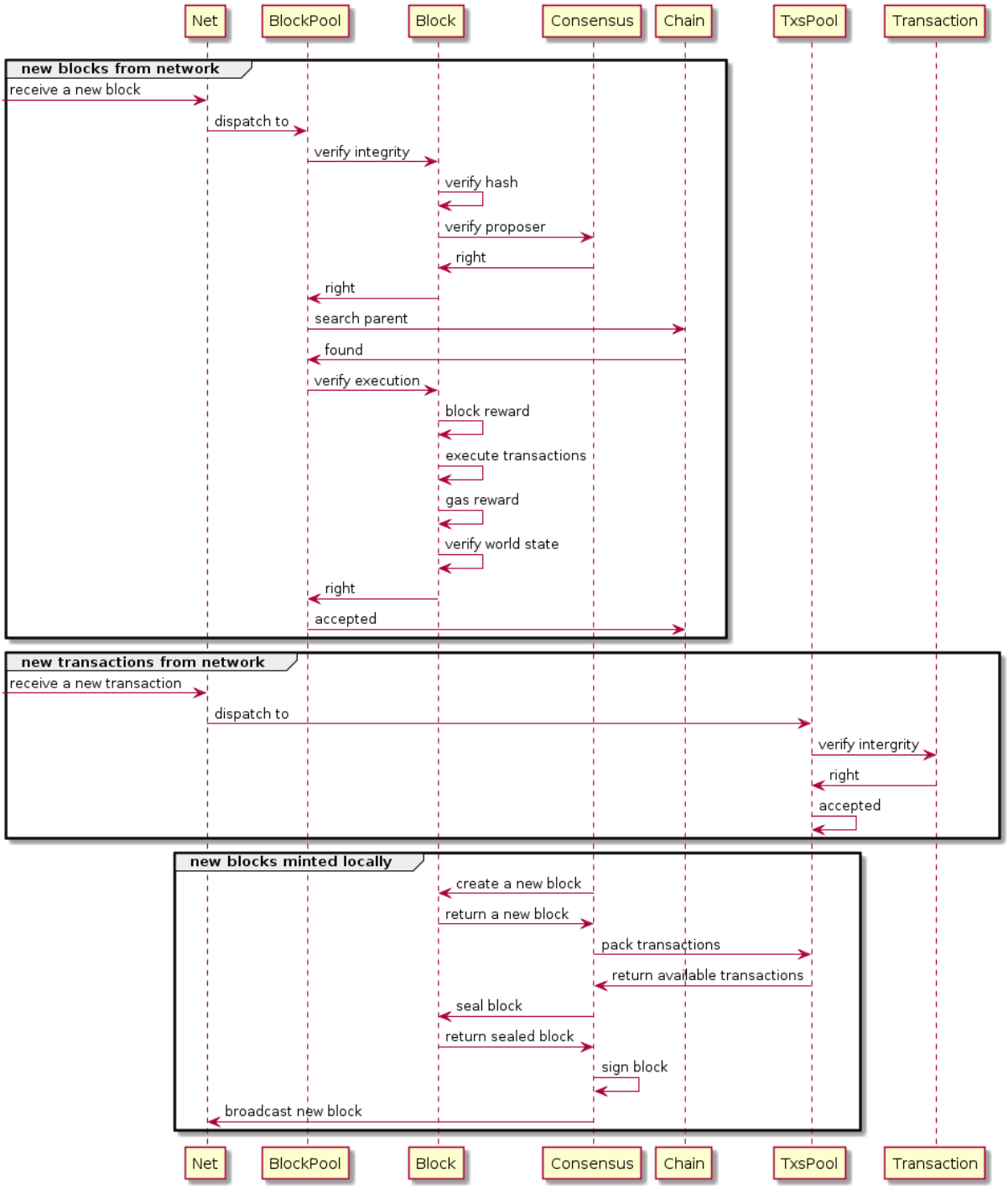
Design Overview



TODO: More features described in our whitepaper, such as NR, PoD, DIP and NF, will be integrated into the framework in later versions very soon.

Core Dataflow

Here is a core workflow example to explain how Nebulas works in current version. For each Nebulas node, it keeps receiving blocks or transactions from network and mining new block locally.



More Details

Blockchain

Model

Nebulas use accounts model instead of UTXO model. The execution of transactions will consume gas.

Data Structure

```
Block Structure
+---------------+----------------+--------------+
|  blockHeader  |  transactions  |  dependency  |
+---------------+----------------+--------------+
blockHeader: header info
transactions: transactions array
dependency: the dependency relationship among transactions

Block Header Structure
+-----------+--------+--------------+------------+-------------+----
 ↪---+--------+
|  chainid  |  hash  |  parentHash  |  coinbase  |  timestamp  | ␣
 ↪alg  |  sign  |
+-----------+--------+--------------+------------+-------------+----
 ↪---+--------+
+-------------+-----------+--------------+-----------------+
|  stateRoot  |  txsRoot  |  eventsRoot  |  consensusRoot  |
+-------------+-----------+--------------+-----------------+
chainid: chain identity the block belongs to
hash: block hash
parentHash: parent block hash
coinbase: account to receive the mint reward
timestamp: the number of nanoseconds elapsed since January 1, 1970␣
 ↪UTC
alg: the type of signature algorithm
sign: the signature of block hash
stateRoot: account state root hash
txsRoot: transactions state root hash
eventsRoot: events state root hash
consensusRoot: consensus state, including proposer and the dynasty␣
 ↪of validators

Transaction Structure
+-----------+--------+--------+------+---------+---------+----------
 ↪---+
|  chainid  |  hash  |  from  |  to  |  value  |  nonce  | ␣
 ↪timestamp  |
+-----------+--------+--------+------+---------+---------+----------
 ↪---+
+--------+------------+------------+
|  data  |  gasPrice  |  gasLimit  |
+--------+------------+------------+
chainid: chain identity the block belongs to
hash: transaction hash
from: sender's wallet address
to: receiver's wallet address
value: transfer value
nonce: transaction nonce
```

```
timestamp: the number of seconds elapsed since January 1, 1970 UTC
alg: the type of signature algorithm
sign: the signature of block hash
data: transaction data, including the type of transaction(binary␣
↪transfer/deploy smart contracts/call smart contracts) and payload
gasPrice: the price of each gas consumed by the transaction
gasLimit: the max gas that can be consumed by the transaction
```

Blockchain Update

In our opinion, **Blockchain** only needs to care about how to process new blocks to grow up safely and efficiently. What's more, **Blockchain** can only get new blocks in the following two channels.

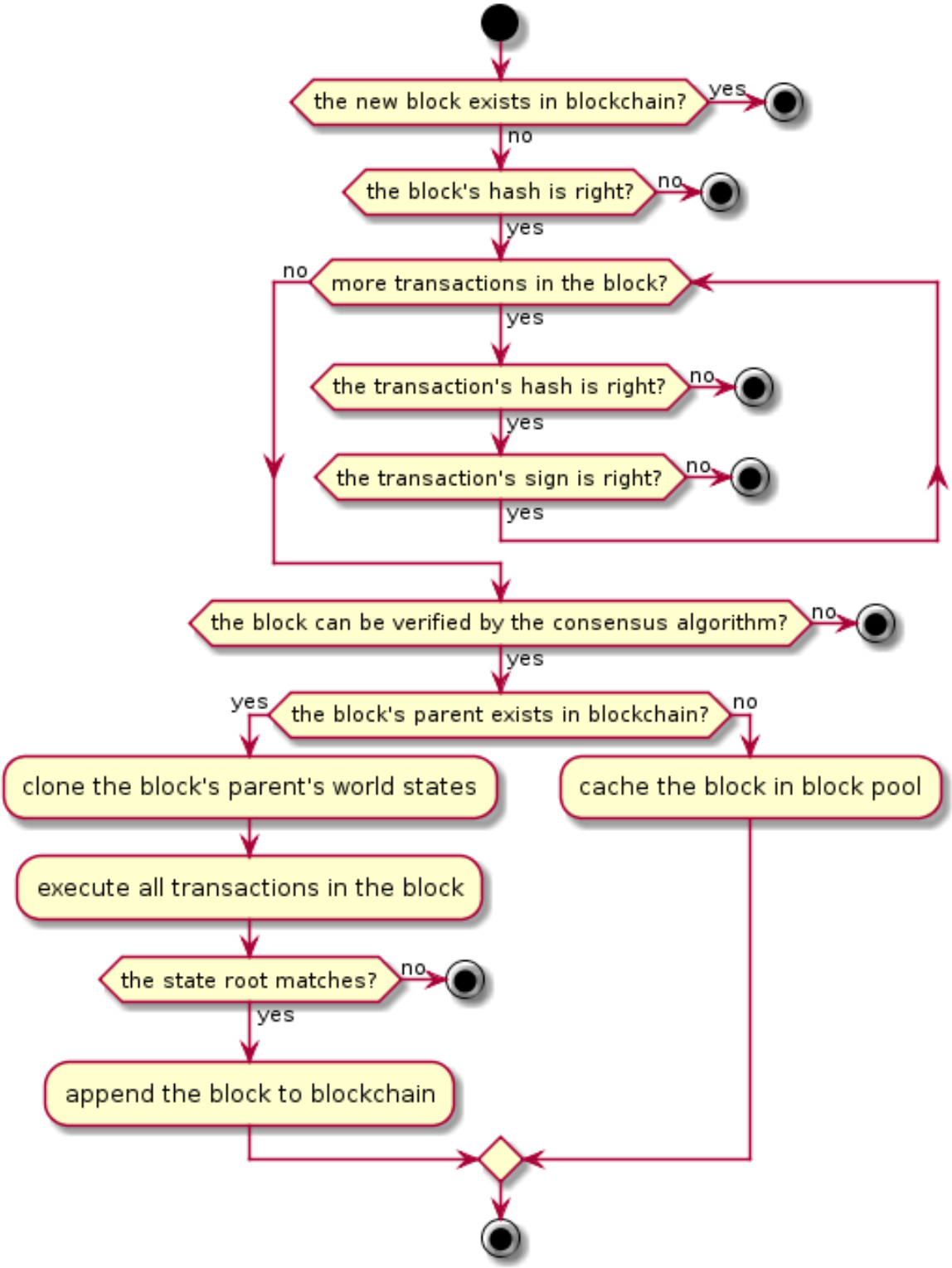
A new block from network

Because of the unstable network latency, we cannot make sure any new block received can be linked to our current **Chain** directly. Thus, we need the **Blocks Pool** to cache new blocks.

A new block from local miner

At first, we need the **Transactions Pool** to cache transactions from network. Then, we wait for a new block created by local **Consensus** component, such as DPoS.

No matter where a new block comes from, we use the same steps to process it as following.



World State

Every block contains the current world state, consist of following four states. They are all maintained as Merkle Trees.

Accounts State

All accounts in current block are stored in Accounts State. Accounts are divided into two kinds, normal account & smart contract account.

Normal Account, including

- **wallet address**
- **balance**
- **nonce**: account‘s nonce, it will increment in steps of 1

Smart Contract AccountïijŇ including

- **contract address**
- **balance**
- **birth place**: the transaction hash where the contract is deployed
- **variables**: contains all variables‘ values in the contract

Transactions State

All transactions submitted on chain are storage in Transactions State.

Events State

While transactions are executed, many events will be triggered. All events triggered by transactions on chain are stored in Events State.

Consensus State

The context of consensus algorithm is stored in consensus state.

As for DPoS, the consensus state includes

- **timestamp**: current slot of timestamp
- **proposer**: current proposer
- **dynasty**: current dynasty of validators

Serialization

We choose Protocol Buffers to do general serialization in consideration of the following benefits:

- Large scale proven.

- Efficiency. It omits key literals and use varints encoding.
- Multi types and multilangue client support. Easy to use API.
- Schema is good format for communication.
- Schema is good for versioning/extension, i.e., adding new message fields or deprecating unused ones.

Specially, we use json to do serialization in smart contract codes instead of protobuf for the sake of readability.

Synchronization

Sometimes we will receive a block with height much higher than its current tail block. When the gap appears, we need to sync blocks from peer nodes to catch up with them.

Nebulas provides two method to sync blocks from peers: Chunks Downloader and Block Downloader. If the gap is bigger than 32 blocks, we'll choose Chunk Downloader to download a lot of blocks in chunks. Otherwise, we choose Block Downloader to download block one by one.

Chunks Downloader

Chunk is a collection of 32 successive blocks. Chunks Downloader allows us to download at most 10 chunks following our current tail block each time. This chunk-based mechanism could help us minimize the number of network packets and achieve better safety.

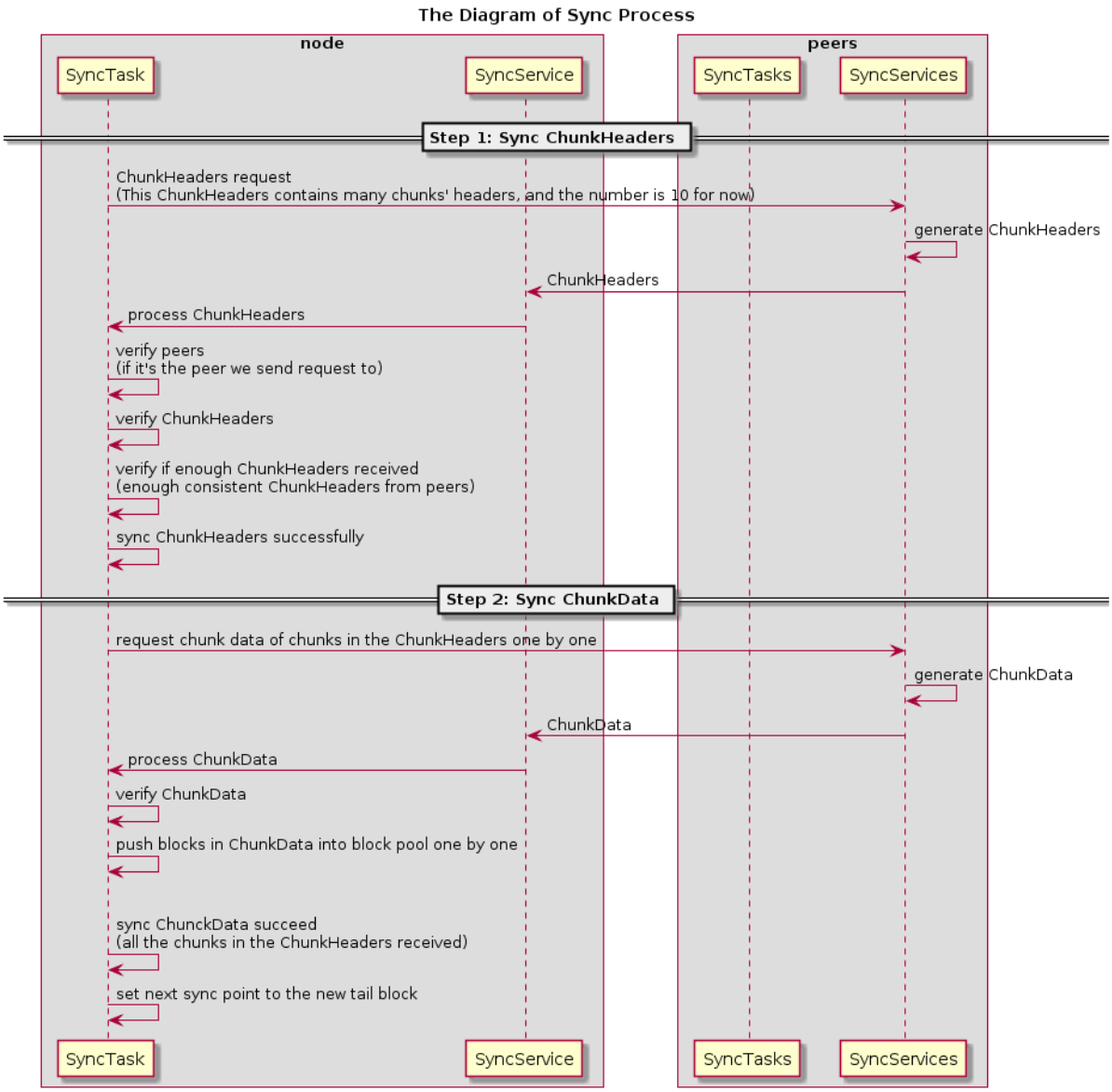
The procedure is as following,

```
1. A sends its tail block to N remote peers.
2. The remote peers locate the chunk C that contains A's tail block.
   Then they will send back the headers of 10 chunks, including the␣
↪chunk C and 9 C's subsequent chunks, and the hash H of the 10␣
↪headers.
3. If A receives >N/2 same hash H, A will try to sync the chunks␣
↪represented by H.
4. If A has fetched all chunks represented by H and linked them on␣
↪chain successfully, Jump to 1.
```

In steps 1~3, we use majority decision to confirm the chunks on canonical chain. Then we download the blocks in the chunks in step 4.

Note: `ChunkHeader` contains an array of 32 block hash and the hash of the array. `ChunkHeaders` contains an array of 10 `ChunkHeaders` and the hash of the array.

Here is a diagram of this sync procedure:



Block Downloader

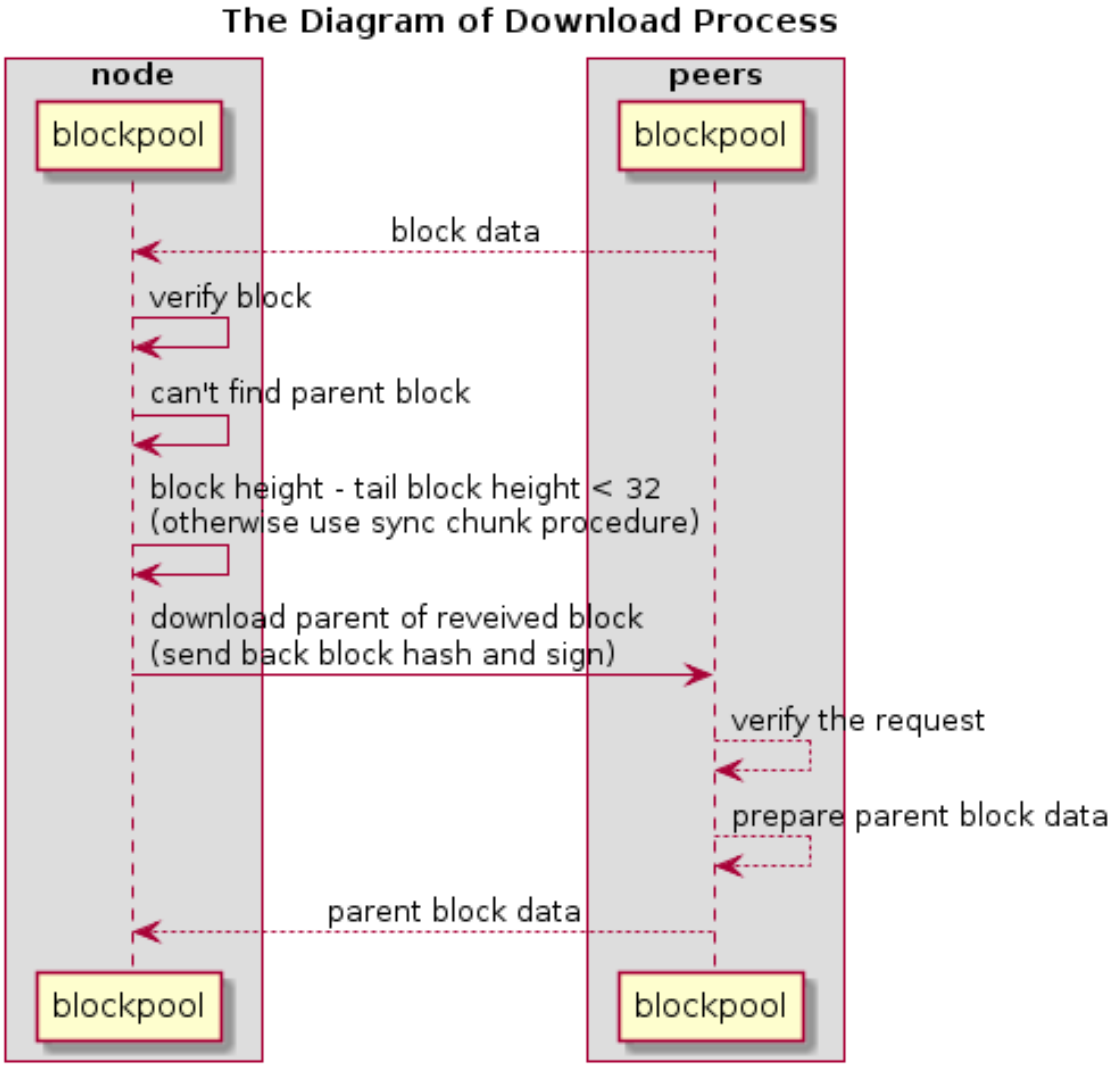
When the length gap between our local chain with the canonical chain is smaller than 32, we‘ll use Block downloader to download the missing blocks one by one.

The procedure is as following,

```
1. C relays the newest block B to A and A finds B's height is␣
↪bigger than current tail block's.
2. A sends the hash of block B back to C to download B's parent␣
↪block.
3. If A received B's parent block B', A will try to link B' with A
↪'s current tail block.
   If failed again, A will come back to step 2 and continue to␣
↪download the parent block of B'. Otherwise, finished.
```

This procedure will repeat until A catch up with the canonical chain.

Here is a diagram of this download procedure:



Merkle Patricia Tree

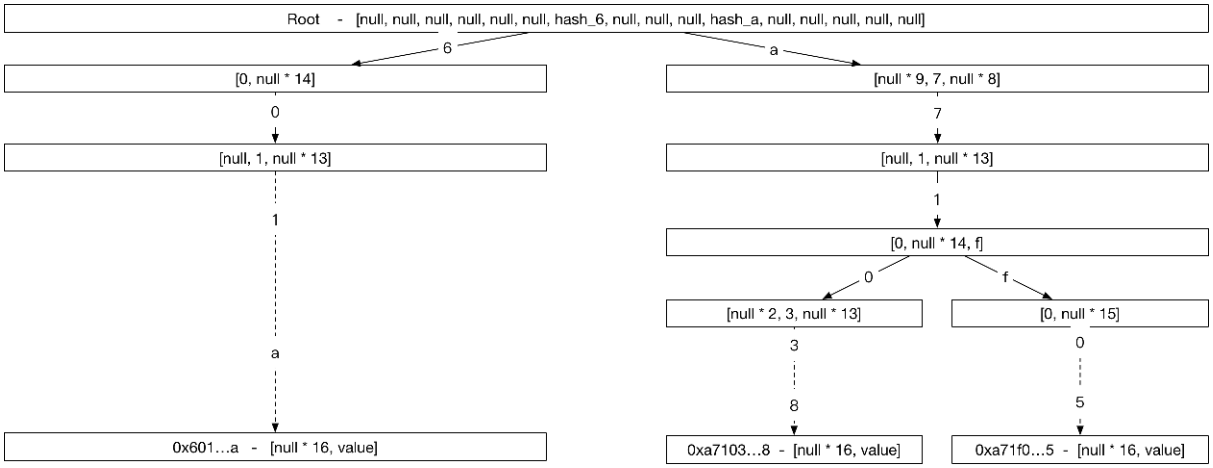
Basic: Radix Tree

Reference: https://en.wikipedia.org/wiki/Radix_tree

A Radix Tree using address as the key looks like below:

- Addresses are represented as Hex Characters
- Each node in the Tree is a 16-elements array, 16 branch-slots(0123...def)
- leaf node: value can be any binary data carried by the address
- non-leaf node: value is the hash value calculated based on the childrenâĂŹs data

As for a 160-bits address, the max height of the tree is 40



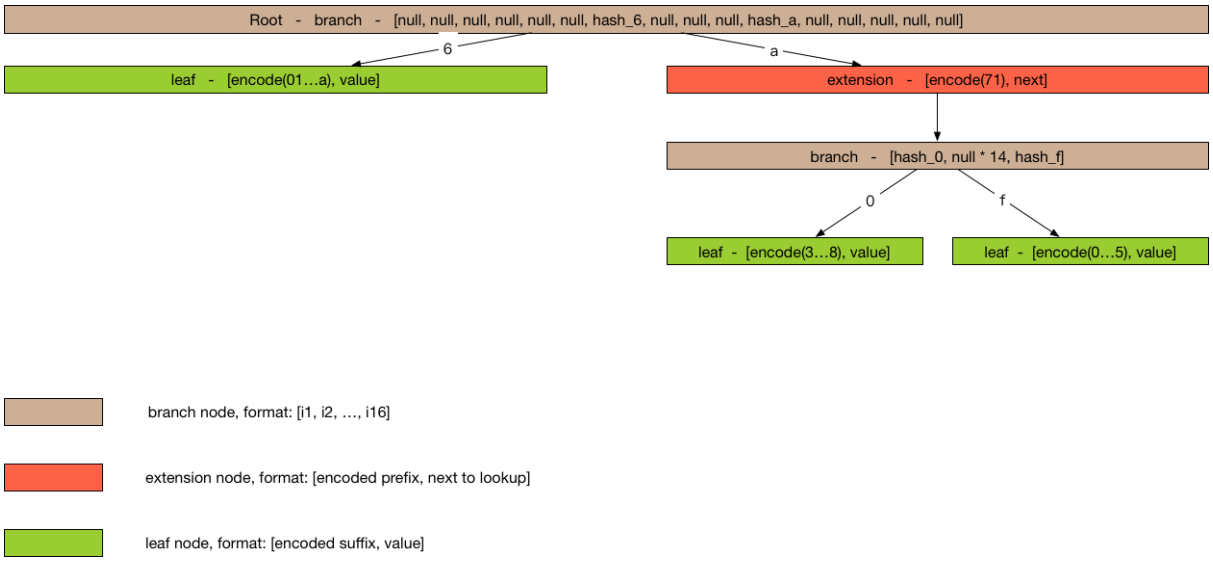
Problems: much space for a single entry 40 steps for each lookup

Advanced: Merkle Patricia Tree

Reference: https://github.com/ethereum/wiki/wiki/Patricia-Tree, http://gavwood.com/Paper.pdf

In order to reduce the storage of Radix Tree. The nodes in Merkle Patricia Tree are divided into three kinds,

- extension node: compress nodes using common prefix
- leaf node: compress nodes using unique suffix
- branch node: same as node in Radix Tree



How to store Merkle Patricia Tree

Key/Value Storage

hash(value) = sha3(serialize(value))

key = hash(value)

How to update Merkle Patricia Tree

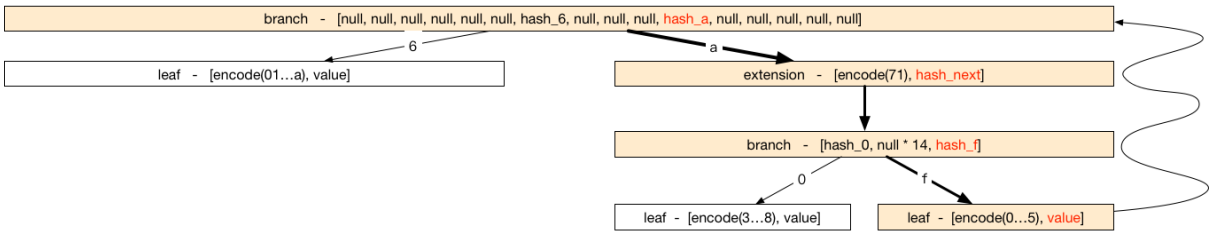
Query

DFS from top to bottom

Update, Delete or Insert

1.Query the node from top to bottom

2.update the hash along the path from bottom to top



Performance Each operation costs O(log(n))

How to verify using Merkle Patricia Tree

Theorems

1.Same merkle trees must have same root hash.

2.Different merkle trees must have different root hash.

Using the theorems, we can verify the result of the execution of transactions.

Quick Verification

A light client, without sync huge transactions, can immediately determine the exact balance and status of any account by simply asking the network for a path from the root to the account node.

Consensus

We think each consensus algorithm can be described as the combination of State Machine and Fork Choice Rules.

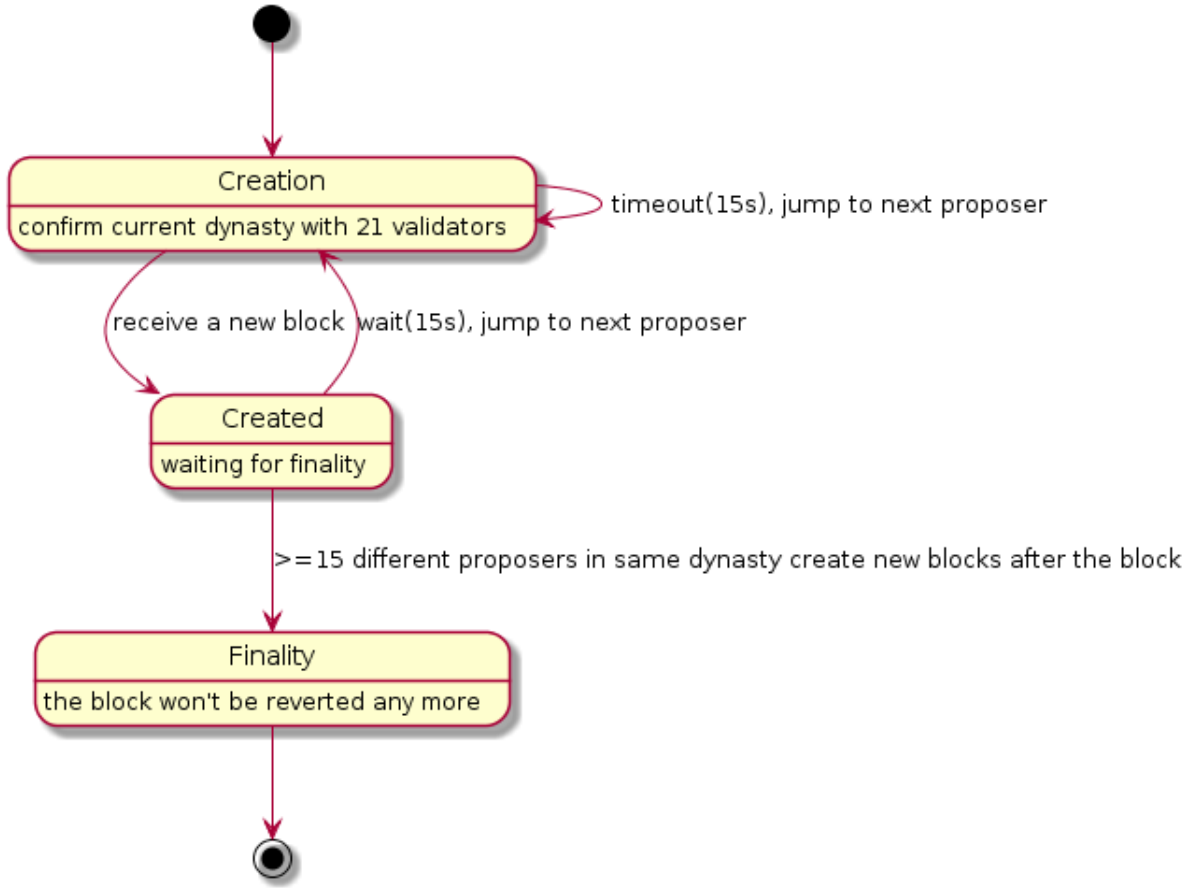
DPoS(Delegate Proof-of-Stake)

Notice For Nebulas, the primary consensus algorithm should be PoD, the DPoS algorithm is just a temporary solution. After the formal verification of PoD algorithm, we will transition mainnet to PoD. All witness (bookkeeper/miner) of DPoS

are now accounts officially maintained by Nebulas. We will make sure a smooth transition from DPoS to PoD. We will create new funds to manage all the rewards of bookkeeping. And we will NOT sell those NAS on exchanges. All NAS will be used for building the Nebulas ecosystem, for example, rewarding DApp developers on Nebulas. And we will provide open access to all the spending of these rewards periodically.

As for the DPoS in Nebulas, it can also be decribed as a state machine.

State Machine



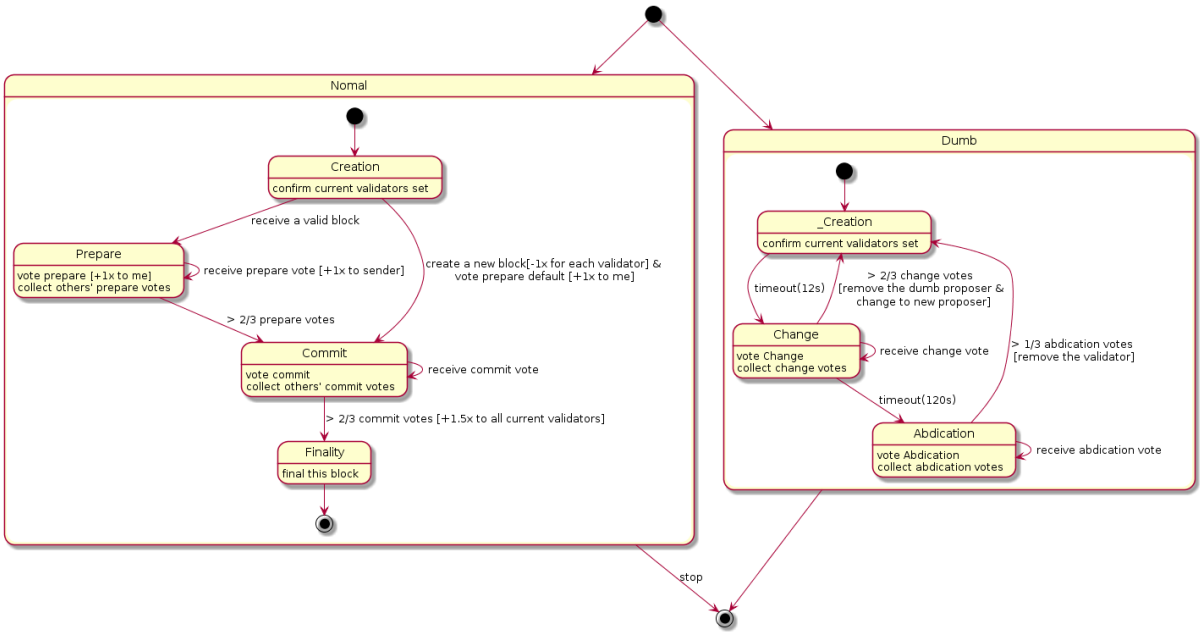
Fork Choice Rules

1. Always choose the longest chain as the canonical chain.
2. If A and B has the same length, we choose the one with smaller hash.

PoD (Proof-of-Devotion)

Here is a draft of PoD. The research on PoD is ongoing here.

State Machine



Fork Choice Rules

1. Always to choose the chain with highest sum of commit votes.
2. If A and B has the same length, we choose the one with smaller hash.

Transaction Process Diagram

When a transaction is submitted, it is necessary to check the chain in the transaction. Transactions that are submitted externally or have been packaged into the block are somewhat different when doing validation.

New Transaction Process (from network, rpc)

Transactions submitted through an RPC or other node broadcast.

- Api SendRawTransaction Verification below steps when exist fail, then return err
- check whether fromAddr and toAddr is valid (tx proto verification)
- check len of Payload <= MaxDataPayLoadLength (tx proto verification)
- 0 < gasPrice <= TransactionMaxGasPrice and 0 < gasLimit <= TransactionMaxGas (tx proto verification)
- check Alg is SECP256K1 (tx proto verification)
- chainID Equals, Hash Equals, Sign verify??; fail and drop;

- check nonceOfTx > nonceOfFrom
- check Contract status is ExecutionSuccess if type of tx is TxPayloadCallType, check toAddr is equal to fromAddr if type of tx is TxPayloadDeployType
- Transaction pool Verification
- gasPrice >= minGasPriceOfTxPool & 0 < gasLimit <= maxGasLimitOfTxPool??; fail and drop;
- chainID Equals, Hash Equals, Sign verify??; fail and drop;

Transaction in Block Process

The transaction has been packaged into the block, and the transaction is verified after receiving the block.

- Packed
- Nonce Verification: nonceOfFrom +1 == nonceOfTx ??; nonceOfTx < nonceOfFrom +1 fail and drop, nonceOfTx > nonceOfFrom +1 fail and giveback to tx pool;
- check balance >= gasLimit * gasPrice ??; fail and drop;
- check gasLimit >= txBaseGas(MinGasCountPerTransaction + dataLen*GasCountPerByte) ??; fail and drop;
- check payload is valid ??; fail and submit; gasConsumed is txBaseGas (all txs passed the step tx will be on chain)
- check gasLimit >= txBaseGas + payloasBaseGas(TxPayloadBaseGasCount[payloadType]) ??;fail and submit; gasConsumed is txGasLimit
- check balance >= gasLimit * gasPrice + value ??;fail and submit; gasConsumed is txBaseGas + payloadsBaseGas
- transfer value from SubBalance and to AddBalance ??;fail and submit; gasConsumed is txBaseGas + payloadsBaseGas
- check gasLimit >= txBaseGas + payloadsBaseGas + gasExecution ??;fail and submit; gasConsumed is txGasLimit
- success submit gasConsumed is txBaseGas + payloadsBaseGas + gasExecution
- Verify
- check whether fromAddr and toAddr is valid (tx proto verification) ??; fail and submit;
- check len of Payload <= MaxDataPayLoadLength (tx proto verification) ??; fail and submit;
- 0 < gasPrice <= TransactionMaxGasPrice and 0 < gasLimit <= TransactionMaxGas (tx proto verification)
- check Alg is SECP256K1 (tx proto verification) ??; fail and submit;
- chainID Equals, Hash Equals, Sign verify??; fail and drop;

- Next steps like Transaction Packed in Block Process.

Event functionality

The `Event` functionality is used to make users or developers subscribe interested events. These events are generated during the execution of the blockchain, and they record the key execution steps and execution results of the chain. To query and verify the execution results of transactions and smart contracts, we record these two types of events into a trie and save them to the chain.

Event structure:

```
type Event struct {
    Topic string // event topic, subscribe keyword
    Data  string // event content, a json string
}
```

After a event is generated, it will be collected for processing in eventEmitter. Users can use the emitter subscription event. If the event is not subscribed, it will be discarded, and for the event that has been subscribed, the new event will be discarded because of the non-blocking mechanism, if the channel is not blocked in time.

Events list:

- TopicNewTailBlock
- TopicRevertBlock
- TopicLibBlock
- TopicPendingTransaction
- TopicTransactionExecutionResult
- EventNameSpaceContract

Event Reference

TopicNewTailBlock

This event occurs when the tail block of the chain is updated.

- Topic:`chain.newTailBlock`
- Data:
  - `height`: block height
  - `hash`: block hash
  - `parent_hash`: block parent hash

  - `acc_root`: account state root hash
  - `timestamp`: block timestamp
  - `tx`: transaction state root hash
  - `miner`: block miner

TopicRevertBlock

This event occurs when a block is revert on the chain.

- Topic:`chain.revertBlock`
- Data: The content of this topic is like TopicNewTailBlock data.

TopicLibBlock

This event occurs when the latest irreversible block change.

- Topic:`chain.latestIrreversibleBlock`
- Data: The content of this topic is like TopicNewTailBlock data.

TopicPendingTransaction

This event occurs when a transaction is pushed into the transaction pool.

- Topic:`chain.pendingTransaction`
- Data:
  - `chainID`: transaction chain id
  - `hash`: transaction hash
  - `from`: transaction from address string
  - `to`: transaction to address string
  - `nonce`: transaction nonce
  - `value`: transaction value
  - `timestamp`: transaction timestamp
  - `gasprice`: transaction gas price
  - `gaslimit`: transaction gas limit
  - `type`: trsnaction type

TopicTransactionExecutionResult

This event occurs when the end of a transaction is executed. This event will be recorded on the chain, and users can query with RPC interface GetEventsByHash.

This event records the execution results of the transaction and is very important.

- Topic:`chain.transactionResult`
- Data:
  - `hash`: transaction hash
  - `status`: transaction status, 0 failed, 1success, 2 pending
  - `gasUsed`: transaction gas used
  - `error`: transaction execution error. If the transaction is executed successfully, the field is empty.

EventNameSpaceContract

This event occurs when the contract is executed. When the contract is executed, the contract can record several events in the execution process. If the contract is successful, these events will be recorded on the chain and can be subscribed, and the event of the contract will not be recorded at the time of the failure. This event will also be recorded on the chain, and users can query with RPC interface GetEventsByHash.

- Topic:`chain.contract.[topic]` The topic of the contract event has a prefix `chain.contract.`, the content is defined by the contract writer.
- Data: The content of contract event is defined by contract writer.

Subscribe

All events can be subscribed and the cloud chain provides a subscription RPC interface Subscribe. It should be noted that the event subscription is a non-blocking mechanism. New events will be discarded when the RPC interface is not handled in time.

Query

Only events recorded on the chain can be queried using the RPC interface GetEventsByHash. Current events that can be queried include:

- TopicTransactionExecutionResult
- EventNameSpaceContract

Transaction Gas

In Nebulas, either a normal transaction which transfer balance or a smart contract deploy & call burns gas, and charged from the balance of `from` address. A transaction contains two gas parameters `gasPrice` and `gasLimit` :

- `gasPrice`: the price of per gas.
- `gasLimit`: the limit of gas use.

The actual gas consumption of a transaction is the value: `gasPrice` * `gasUsed`, which will be the reward to the miner coinbase. The `gasUsed` value must less than or equal to the `gasLimit`. Transaction‘s `gasUsed` can be estimate by RPC interface estimategas and store in transaction‘s execution result event.

Design reason

Users want to avoid gas costs when the transaction is packaged. Like Bitcoin and Ethereum, Nebulas GAS is used for transaction fee, it have two major purposes:

- As a rewards for minter, to incentive them to pack transactions. The packaging of the transaction costs the computing resources, especially the execution of the contract, so the user needs to pay for the transaction.
- As a cost for attackers. The DDOS attach is quite cheap in Internet, black hackers hijack user‘s computer to send large network volume to target server. In Bitcoin and Ethereum network, each transaction must be paid, that significant raise the cost of attack.

Gas constitution

When users submit a transaction, gas will be burned at these aspects:

- `transaction submition`
- `transaction data storage`
- `transaction payload addition`
- `transaction payload execution`(smart contract execution)

In all these aspects, the power and resources of the net will be consumed and the miners will need to be paid.

Transaction submition

A transaction‘s submition will add a transaction to the tail block. Miners use resources to record the deal and need to be paid. It will burn a fixed number of gas, that would be defined in code as the following:

```
// TransactionGas default gas for normal transaction
TransactionGas = 20000
```

If the transaction verifies failed, the gas and value transfer will rollback.

Transaction data storage

When deploying a contract or call contract‘s method, the raw data of contract execution save in the transaction‘s data filed, which cost the storage of resources on the chain. A formula to calculate gas:

```
TransactionDataGas = 1

len(data) * TransactionDataGas
```

The `TransactionDataGas` is a fixed number of gas defined in code.

Different types of transactions‘ payload have different gas consumption when executed. The types of transactions currently supported by nebulas are as follows:

- `binary`: The `binary` type of transaction allows users to attach binary data to transaction execution. These binary data do not do any processing when the transaction is executed.
  - The fixed number of gas defined **0**.
- `deploy & call`: The `deploy` and `call` type of transaction allows users to deploy smart contract on nebulas. Nebulas must start `nvm` to execute the contract, so these types of transction must paid for the nvm start.
  - The fixed number of gas defined **60**.

Transaction payload execution(Smart contract deploy & call)

The `binary` type of transaction do not do any processing when the transaction is executed, so the execution need not be paid.

When a smart contract deploys or call in transaction submition, the contract execution will consume miner‘s computer resources and may store data on the chain.

- **execution instructions**: Every contract execution cost the miner‘s computer resources, the v8 instruction counter calculates the execution instructions. The limit of execution instructions will prevent the excessive consumption of computer computing power and the generation of the death cycle.
- **contract storage**: The smart contract‘s `LocalContractStorage` which storage contract objects also burn gas. Only one gas per 32 bytes is consumed when stored(`set`/`put`), `get` or `delete` not burns gas.

The limit of **contract execution** is:

```
    gasLimit - TransactionGas - len(data) * TransactionDataGas -␣
↪TransactionPayloadGasCount[type]
```

Gas Count Matrix

The gas count matrix of smart contract execution

| Expression | Sample Code | Binary Opt. | Load Opt. | Store Opt. | Return Opt. | Call (inner) Opt. | Gas Count | | | | — | :— | —: | —: | —: | —: | —: | | CallExpression | a(x, y) | 0 | 0 | 1 | 1 | 1 | 8 | | | | AssignmentExpression | x&=y | 1 | 0 | 1 | 0 | 0 | 3 | | | | BinaryExpression | x==y | 1 | 0 | 0 | 1 | 0 | 3 | | | | UpdateExpression | x++ | 1 | 0 | 1 | 0 | 0 | 3 | | | | UnaryExpression | x+y | 1 | 0 | 0 | 1 | 0 | 3 | | | | LogicalExpression | x | | y | 1 | 0 | 0 | 1 | 0 | 3 | | MemberExpression | x.y | 0 | 1 | 0 | 1 | 0 | 4 | | | | NewExpression | new X() | 0 | 0 | 1 | 1 | 1 | 8 | | | | ThrowStatement | throw x | 0 | 0 | 0 | 1 | 1 | 6 | | | | MetaProperty | new.target | 0 | 1 | 0 | 1 | 0 | 4 | | | | ConditionalExpression | x?y:z | 1 | 0 | 0 | 1 | 0 | 3 | | | | YieldExpression | yield x | 0 | 0 | 0 | 1 | 1 | 6 | | | | Event | | 0 | 0 | 0 | 0 | 0 | 20 | | | | Storage | | 0 | 0 | 0 | 0 | 0 | 1 gas/bit | | |

Tips

In nebulas, the transaction pool of each node has a minimum and maximum `gasPrice` and maximum `gasLimit` value. If transaction‘s `gasPrice` is not in the range of the pool‘s `gasPrice` or the `gasLimit` greater than the pool‘s gasLimit the transaction will be refused.

Transaction pool gasPrice and gasLimit configuration:

- `gasPrice`
  - minimum: The minimum gasPrice can be set in the configuration file. If the minimum value is not configured, the default value is `20000000000`(2*10^10).
  - maximum: The maximum gasPrice is `1000000000000`(10^12), transaction pool‘s maximum configuration and transaction‘s `gasPrice` can‘t be overflow.
- `gasLimit`
  - minimum: The transaction‘s minimum gasLimit must greater than zero.
  - maximum: The maximum gasPrice is `50000000000`(50*10^9), transaction pool‘s maximum configuration and transaction‘s `gasLimit` can‘t be overflow.

Logs

Introduction

Nebulas provides two kinds of logs: console log & verbose log.

Console Log

Console Log(CLog) is used to help you understand which job **Neb** is working on now, including start/stop components, receive new blocks on chain, do synchronization and so on.

- CLog will print all logs to stdout & log files both. You can check them in your standard output directly.

Nebulas console log statements

```
// log level can be `Info`,`Warning`,`Error`
logging.CLog().Info("")
```

Startup specifications

Nebulas start service should give a console log, the logs should before the service start. The log format just like this:

```
logging.CLog().Info("Starting xxx...")
```

Stopping specifications

Nebulas stop service should give a console log, the logs should before the service stoped. The log format just like this:

```
logging.CLog().Info("Stopping xxx...")
```

Verbose Log

Verbose Log(VLog) is used to help you understant how **Neb** works on current job, including how to verifiy new blocks, how to discover new nodes, how to mint and so on.

- VLog will print logs to log files only. You can check them in your log folders if needed.

What‘r more, you can set your concerned level to VLog to filter informations. The level filter follows the priority as **Debug < Info < Warn < Error < Fatal.**

Hookers

By default, Function hookers & FileRotate hookers are added to CLog & VLog both.

FunctionNameHooker

FunctionHooker will append current caller‘s function name & code line to the loggers. The result looks like this,

time="2018-01-03T20:20:52+08:00" level=info msg="node init success" file=net_service.go **func=p2p.NewNetManager** **line=137** node.listen="[0.0.0.0:10001]"

FileRotateHooker

FileRotateHooker will split logs into many smaller segments by time. By default, all logs will be rotated every 1 hour. The log folder looks like this,

neb-2018010415.log neb-2018010416.log neb.log -> /path/to/neb-2018010415.log

If you have any suggestions about logs, please feel free to submit issues on our wiki repo. Thanks!

Nebulas Address Design

Nebulas address system is carefully designed. As you will see below, both account and smart contract address are strings starting with a "n", which could be thought of as our faith Nebulas/NAS.

Account Address

Similar to Bitcoin and Ethereum, Nebulas also adopts elliptic curve algorithm as its basic encryption algorithm for Nebulas accounts. The address is derived from **public key**, which is in turn derived from the **private key** that encrypted with user's **passphrase**.Also we have the checksum design aiming to prevent a user from sending *Nas* to a wrong user account accidentally due to entry of several incorrect characters.

The specific calculation formula is as follows:

```
1.  content = ripemd160(sha3_256(public key))
    length: 20 bytes
                          +--------+--------+------------------+
2.  checksum = sha3_256( |  0x19  +  0x57  |      content     |␣
↪)[:4]
                          +--------+--------+------------------+
    length: 4 bytes

                         +--------+---------+-----------------+------
↪------+
3.  address = base58( |    0x19  |  0x57   |     content     | ␣
↪checksum  | ïijL'
                         +--------+---------+-----------------+------
↪------+
    length: 35 chars
```

0x57 is a one-byte "type code" for account address, **0x19** is a one-byte fixed "padding"

At this stage, Nebulas just adopts the normal bitcoin base58 encoding schema. A valid address is like: *n1TV3sU6jyzR4rJ1D7jCAmtVGSntJagXZHC*

Smart Contract Address

Calculating contract address differs slightly from account, passphrase of contract sender is not required but address & nonce. For more information, please check smart contract and rpc.sendTransaction. Calculation formula is as follows:

```
1.  content = ripemd160(sha3_256(tx.from, tx.nonce))
    length: 20 bytes
                            +--------+--------+------------------+
2.  checksum = sha3_256( |  0x19  |  0x58  +      content     |␣
↪)[:4]
                            +--------+--------+------------------+
    length: 4 bytes

                        +--------+---------+-----------------+--------
↪----+
3.  address = base58( |  0x19  |  0x58   |     content     | ␣
↪checksum  | ïijL'
                        +--------+---------+-----------------+--------
↪----+
    length: 35 chars
```

0x58 is a one-byte "type code" for smart contract address, **0x19** is a one-byte fixed "padding"

A valid address is like: *n1sLnoc7j57YfzAVP8tJ3yK5a2i56QrTDdK*

DIP (TBD)

Como se Juntar Ãă Mainnet da Nebulas

IntroduÃğÃčo

A Mainnet 2.0 da Nebulas (Nebulas Nova) foi lanÃğada. Este tutorial visa explicar como utilizÃą-la. Junte-se Ãă Mainnet e desfrute!

https://github.com/nebulasio/go-nebulas/tree/master

CompilaÃğÃčo

O ficheiro executÃąvel da Mainnet da Nebulas e as suas bibliotecas tÃłm de ser compilados primeiro. Modulos de alta importÃćncia estÃčo listados abaixo:

- **NBRE:** O Ambiente de ExecuÃğÃčo da Nebulas Ãl' a plataforma que corre o Protocolo de RepresentaÃğÃčo da Nebulas, como o DIP, o NR, etcetera.

- **NEB:** O processo principal da Mainnet da Nebulas, o `NEB` e o `NBRE` correm em processos individuais, e comunicam atravÃl's de IPC

Detalhes sobre a compilaÃğÃčo dos modulos pode ser encontrada em tutoriais.

ConfiguraÃğÃčo

Os ficheiros de configuraÃğÃčo da Mainnet estÃčo no directÃşrio `mainnet/conf`, incluÃ■ndo

genesis.conf

Toda a informaÃğÃčo da configuraÃğÃčo do bloco genesis estÃą definida em genesis.conf, incluÃ■ndo

- **meta.chain_id:** cadeia de identidade
- **consensus.dpos.dynasty:** a dinastia inicial dos validadores
- **token_distribution:** a alocaÃğÃčo inicial de tokens

AtenÃğÃčo: NÃČO altere o genesis.conf.

config.conf

Toda a informaÃğÃčo da configuraÃğÃčo sobre o runtime estÃą definida em config.conf.

Por favor, verifique o `template.conf` para encontrar mais detalhes sobre a configuraÃğÃčo do tempo de execuÃğÃčo.

Nota: a informaÃğÃčo do nÃş da raÃ■z oficial estÃą descrita em baixo,

```
seed:["/ip4/52.2.205.12/tcp/8680/ipfs/
↪QmQK7W8wrByJ6So7rf84sZzKBxMYmc1i4a7JZsne93ysz5","/ip4/52.56.55.
↪238/tcp/8680/ipfs/QmVy9AHxBpd1iTvECDR7fvdZnqXeDhnxkZJrKsyuHNYKAh",
↪"/ip4/13.251.33.39/tcp/8680/ipfs/
↪QmVm5CECJdPAHmzJWN2X7tP335L5LguGb9QLQ78riA9gw3"]
```

Lista do API

Ponto final principal:

- GetNebState : retorna a informaÃğÃčo do cliente Nebulas.
- GetAccountState: retorna o saldo da conta e o nonce.
- Call: executa o smart contract local, sem o submeter na chain.
- SendRawTransaction: submete a transaÃğÃčo assinada.

- GetTransactionReceipt: obtÃl'm a factura da transacÃğÃčo atravÃl's da sua hash.

Mais APIs Nebulas em RPC.

Tutoriais

PortuguÃłs

1. InstalaÃğÃčo (crÃl'dito: Cristiano)
2. Envio de TransacÃğÃţes (crÃl'dito: Cristiano)
3. Criar Smart Contract em JavaScript (crÃl'dito: Cristiano)
4. IntroduÃğÃčo de Armazenamento em Smart Contracts (crÃl'dito: Cristiano)
5. InteracÃğÃčo com Nebulas atravÃl's do API RPC (crÃl'dito: Cristiano)

ContribuiÃğÃčo

Sinta-se livre de se juntar Ãă Mainnet Nebulas.ÂăSe encontrar algo errado, submeta um relatÃşrioÂăouÂăenvie uma pull requestÂăpara nos informar, e adicionaremos seu nome e seu URL a esta pÃągina o mais rÃąpido possÃ■vel.

Como se Juntar Ãă Testnet da Nebulas

IntroducÃğÃčo

Estamos contentes por lanÃğar a Testnet da Nebulas. Ela simula a rede Nebulas e a NVM, e permite a interacÃğÃčo com a Nebulas sem o pagamento do custo de gÃąs.

https://github.com/nebulasio/go-nebulas/tree/testnet

CompilaÃğÃčo

O ficheiro executÃąvel da Testnet da Nebulas e as suas bibliotecas tÃłm de ser compilados primeiro. Modulos de alta importÃćncia estÃčo listados abaixo:

- **NBRE:** O Ambiente de ExecuÃğÃčo da Nebulas Ãl' a plataforma que corre o Protocolo de RepresentaÃğÃčo da Nebulas, como o DIP, o NR, etcetera.
- **NEB:** O processo principal da Mainnet da Nebulas, o `NEB` e o `NBRE` correm em processos individuais, e comunicam atravÃl's de IPC

Detalhes sobre a compilaÃğÃčo dos modulos pode ser encontrada em tutoriais.

ConfiguraÃğÃčo

Os ficheiros de configuraÃğÃčo estÃčo no directÃşrio `testnet/conf` na branch `testnet`, inclusive

genesis.conf

Toda a informaÃğÃčo configurÃąvel sobre o genesis block estÃą definida em genesis.conf, inclusive

- **meta.chain_id:** identidade da chain
- **consensus.dpos.dynasty:** dinastia inicial dos validadores
- **token_distribution:** alocaÃğÃčo inicial de tokens

AtenÃğÃčo: NÃČO modifique o ficheiro genesis.conf.

config.conf

Toda a informaÃğÃčo configurÃąvel sobre o runtime estÃą definida em config.conf

Por favor veja o `template.conf` para conhecer mais detalhes sobre a configuraÃğÃčo do runtime.

Dica: a informaÃğÃčo do nÃş-raÃ■z oficial estÃą descrita abaixo,

```
seed:["/ip4/52.60.150.236/tcp/8680/ipfs/
↪QmVJikqWQst13QsgdCLBjgcSWwpAAdZjoExGdvK3r2CNhv"]
```

Lista do API

Test Endpoint:

- GetNebState : retorna informaÃğÃčo do cliente de Nebulas.
- GetAccountState: retorna o balanÃğa e nonce da conta.
- LatestIrreversibleBlock: retorna o Ãžltimo bloco irreverssÃ■vel.
- Call: executa o smart contract localmente. A transacÃğÃčo nÃčo serÃą submetida na chain.
- SendRawTransaction: submete transacÃğÃčo assinada. A transacÃğÃčo tem de ser assinada antes de ser enviada.
- GetTransactionReceipt: receba a factura da transacÃğÃčo atravÃl's da hash da mesma.

Mais APIs Nebulas em RPC.

ObtenÃğÃčo de Tokens

Cada email pode pedir tokens uma vez por dia aqui.

Tutoriais

PortuguÃłs

1. InstalaÃğÃčo (obrigado Cristiano)
2. Envio de TransacÃğÃţes (obrigado Cristiano)
3. Criar Smart Contract em JavaScript (obrigado Cristiano)
4. IntroduÃğÃčo de Armazenamento em Smart Contracts (obrigado Cristiano)
5. InteracÃğÃčo com Nebulas atravÃl's do API RPC (obrigado Cristiano)

Contribuir

NÃčo se iniba e junte-se Ãă Testnet da Nebulas. Se encontrou algo de errado, por favor submeta submeta um pedido ou uma pull request para estarmos a par, e adicionaremos o seu nome e url a esta pÃągina o mais rÃąpido possÃ■vel.

Config

There are four types of configuration files in Nebulas.

- Normal node.
- Miner node.(Miner - related configuration is increased relative to normal nodes)
- Super node.(Some connection limits are higher than normal nodes)
- Sign node. (Do not synchronize information with any node, only do signature and unlock)

Normal node

```
network {
  seed: ["/ip4/13.251.33.39/tcp/8680/ipfs/
↪QmVm5CECJdPAHmzJWN2X7tP335L5LguGb9QLQ78riA9gw3"]
  listen: ["0.0.0.0:8680"]
  private_key: "conf/networkkey"
}

chain {
  chain_id:1
```

```
  datadir: "data.db"
  keydir: "keydir"
  genesis: "conf/genesis.conf"
  signature_ciphers: ["ECC_SECP256K1"]
}

rpc {
    rpc_listen: ["0.0.0.0:8784"]
    http_listen: ["0.0.0.0:8785"]
    http_module: ["api","admin"]
    connection_limits:200
    http_limits:200
}

app {
    log_level: "debug"
    log_file: "logs"
    enable_crash_report: true
}

stats {
    enable_metrics: false
}
```

Miner node

```
network {
  seed: ["/ip4/13.251.33.39/tcp/8680/ipfs/
 ↪QmVm5CECJdPAHmzJWN2X7tP335L5LguGb9QLQ78riA9gw3"]
  listen: ["0.0.0.0:8680"]
  private_key: "conf/networkkey"
}

chain {
  chain_id: 1
  datadir: "data.db"
  keydir: "keydir"
  genesis: "conf/genesis.conf"
  coinbase: "n1EzGmFsVepKduN1U5QFyhLqpzFvM9sRSmG"
  signature_ciphers: ["ECC_SECP256K1"]
  start_mine:true
  miner: "n1PxjEu9sa2nvk9SjSGtJA91nthogZ1FhgY"
  remote_sign_server: "127.0.0.1:8694"
  enable_remote_sign_server: true
}

rpc {
    rpc_listen: ["127.0.0.1:8684"]
```

```
    http_listen: ["0.0.0.0:8685"]
    http_module: ["api","admin"]
    connection_limits:200
    http_limits:200
}

app {
    log_level: "debug"
    log_file: "logs"
    enable_crash_report: true
}

stats {
    enable_metrics: false
}
```

Super node

```
network {
  seed: ["/ip4/13.251.33.39/tcp/8680/ipfs/
 ↪QmVm5CECJdPAHmzJWN2X7tP335L5LguGb9QLQ78riA9gw3"]
  listen: ["0.0.0.0:8680"]
  private_key: "conf/networkkey"
  stream_limits: 500
  reserved_stream_limits: 50
}

chain {
  chain_id:1
  datadir: "data.db"
  keydir: "keydir"
  genesis: "conf/genesis.conf"
  signature_ciphers: ["ECC_SECP256K1"]
}

rpc {
    rpc_listen: ["0.0.0.0:8684"]
    http_listen: ["0.0.0.0:8685"]
    http_module: ["api"]
    connection_limits:500
    http_limits:500
    http_cors: ["*"]
}

app {
    log_level: "debug"
    log_file: "logs"
    enable_crash_report: true
```

```
    pprof:{
        http_listen: "0.0.0.0:8888"
    }
}

stats {
    enable_metrics: false
}
```

Sign node

```
network {
  listen: ["0.0.0.0:8680"]
  private_key: "conf/networkkey"
}

chain {
  chain_id:0
  datadir: "data.db"
  keydir: "keydir"
  genesis: "conf/genesis.conf"
  signature_ciphers: ["ECC_SECP256K1"]
}

rpc {
    rpc_listen: ["0.0.0.0:8684"]
    http_listen: ["127.0.0.1:8685"]
    http_module: ["admin"]
    connection_limits:200
    http_limits:200
}

app {
    log_level: "debug"
    log_file: "logs"
    enable_crash_report: true
    pprof:{
        http_listen: "127.0.0.1:8888"
    }
}

stats {
    enable_metrics: false
}
```

How to Develop

Contribution Guideline

The go-nebulas project welcomes all contributors. The process of contributing to the Go project may be different than many projects you are used to. This document is intended as a guide to help you through the contribution process. This guide assumes you have a basic understanding of Git and Go.

Becoming a contributor

Before you can contribute to the go-nebulas project you need to setup a few prerequisites.

Contributor License Agreement

TBD.

Preparing a Development Environment for Contributing

Setting up dependent tools

1. Go dependency management tool

dep is an (not-yet) official dependency management tool for Go. go-nebulas project use it to management all dependencies.

For more information, please visit https://github.com/golang/dep

2. Linter for Go source code

Golint is official linter for Go source code. Every Go source file in go-nebulas must be satisfied the style guideline. The mechanically checkable items in style guideline are listed in Effective Go and the CodeReviewComments wiki page.

For more information about Golint, please visit https://github.com/golang/lint.

3. XUnit output for Go Test

Go2xunit could convert go test output to XUnit compatible XML output used in Jenkins/Hudson.

Making a Contribution

Discuss your design

The project welcomes submissions but please let everyone know what you‘re working on if you want to change or add to the go-nebulas project.

Before undertaking to write something new for the go-nebulas, please file an issue (or claim an existing issue). Significant changes must go through the change proposal process before they can be accepted.

This process gives everyone a chance to validate the design, helps prevent duplication of effort, and ensures that the idea fits inside the goals for the language and tools. It also checks that the design is sound before code is written; the code review tool is not the place for high-level discussions.

Besides that, you can have an instant discussion with core developers in **developers** channel of Nebulas.IO on Slack.

Making a change

Getting Go Source

First you need to fork and have a local copy of the source checked out from the forked repository.

You should checkout the go-nebulas source repo inside your $GOPATH. Go to $GOPATH run the following command in a terminal.

```
$ mkdir -p src/github.com/nebulasio
$ cd src/github.com/nebulasio
$ git clone git@github.com:{your_github_id}/go-nebulas.git
$ cd go-nebulas
```

Contributing to the main repo

Most Go installations project use a release branch, but new changes should only be made based on the **develop** branch. (They may be applied later to a release branch as part of the release process, but most contributors won‘t do this themselves.) Before making a change, make sure you start on the **develop** branch:

```
$ git checkout develop
$ git pull
```

Make your changes

The entire checked-out tree is editable. Make your changes as you see fit ensuring that you create appropriate tests along with your changes. Test your changes as you go.

Copyright

Files in the go-nebulas repository don't list author names, both to avoid clutter and to avoid having to keep the lists up to date. Instead, your name will appear in the change log and in the CONTRIBUTORS file and perhaps the AUTHORS file. These files are automatically generated from the commit logs perodically. The AUTHORS file defines who âĂIJThe go-nebulas AuthorsâĂİâĂŤthe copyright holdersâĂŤare.

New files that you contribute should use the standard copyright header:

```
// Copyright (C) 2017 go-nebulas authors
//
// This file is part of the go-nebulas library.
//
// the go-nebulas library is free software: you can redistribute it␣
↪and/or modify
// it under the terms of the GNU General Public License as␣
↪published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.
//
// the go-nebulas library is distributed in the hope that it will␣
↪be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License
// along with the go-nebulas library.  If not, see <http://www.gnu.
↪org/licenses/>.
//
```

Files in the repository are copyright the year they are added. Do not update the copyright year on files that you change.

Goimports, Golint and Govet

Every Go source file in go-nebulas must pass Goimports, Golint and Govet check. Golint check the style mistakes, we should fix all style mistakes, including comments/docs. Govet reports suspicious constructs, we should fix all issues as well.

Run following command to check your code:

```
$ make fmt lint vet
```

lint.report text file is the Golint report, **vet.report** text file is the Govet report.

Testing

You've written test code, tested your code before sending code out for review, run all the tests for the whole tree to make sure the changes don't break other packages or programs:

```
$ make test
```

test.report text file or **test.report.xml** XML file is the testing report.

Commit your changes

The most importance of committing changes is the commit message. Git will open an editor for a commit message. The file will look like:

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch foo
# Changes not staged for commit:
#    modified:   editedfile.go
#
```

At the beginning of this file is a blank line; replace it with a thorough description of your change. The first line of the change description is conventionally a one-line summary of the change, prefixed by the primary affected package, and is used as the subject for code review email. It should complete the sentence "This change modifies Go to _." The rest of the description elaborates and should provide context for the change and explain what it does. Write in complete sentences with correct punctuation, just like for your comments in Go. If there is a helpful reference, mention it here. If you've fixed an issue, reference it by number with a # before it.

After editing, the template might now read:

```
math: improve Sin, Cos and Tan precision for very large arguments

The existing implementation has poor numerical properties for
large arguments, so use the McGillicutty algorithm to improve
accuracy above 1e10.

The algorithm is described at http://wikipedia.org/wiki/
 ↪McGillicutty_Algorithm

Fixes #159

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch foo
# Changes not staged for commit:
#    modified:   editedfile.go
#
```

The commented section of the file lists all the modified files in your client. It is best to keep unrelated changes in different commits, so if you see a file listed that should not be included, abort the command and move that file to a different branch.

The special notation "Fixes #159" associates the change with issue 159 in the go-nebulas issue tracker. When this change is eventually applied, the issue tracker will automatically mark the issue as fixed. (There are several such conventions, described in detail in the GitHub Issue Tracker documentation.)

Creating a Pull Request

For more information about creating a pull request, please refer to the Create a Pull Request in Github page.

How to debug Go-Nebulas project

ä¡IJèĂĚïijŽWenbo Liu aries.lwb@gmail.com, July 17, 2017

Go-NebulaséąźçŻőåIJřåİĂïijŽhttps://github.com/nebulasio/go-nebulas.git

çőĂäżŃ

è£ŹçґĞç§■æŰĞå§žäžŐMac OSX åŠŇ Ubuntuçşżçż§ïijŇçőĂå■ŢäżŃçż■åęĆä¡ŢèřČèŕŢGo-NebulaséąźçŻőïijŇäÿżèęĄäżŃçż■äÿL'çğ■æŰźæşŢèřČèŕŢïijŽdlvåŚ¡äžd'èąŇèřČèŕŢïijŇGogland IDEèřČèŕŢïijŇäžěåŘŁVisual Studio CodeèřČèŕŢãĂĆ

èřČèŕŢåŹÍDelveåőL'èčĚ

åIJÍ Mac OSX äÿŁåőL'èčĚDelve

GoogleåőŸæŰźäÿžgolangçŽĎèřČèŕŢä¿Ńå■ŘçŤÍgdbïijŇä¡ĘæŸŕdelveæŸŕæŻt'åŘĹéĂĆçŽĎèřČèŕŢåŹÍïij NebulasãĂĆæŹőéĂŽçŽĎgoéąźçŻőæŸŕåŘŕäžěçŽĎïijŇåĚůä¡Şä¡ŞçŐřåřśæŸŕèřČèŕŢGo-NebulaséąźçŻőæŮűïijŇæŰ■çĆźæŮăæşŢåĄIJä¡ŔïijŇäijŽæřÿè£IJhangä¡ŔãĂĆæĹŚäžňå£ĚéążäżőgithubäÿŁä binaryïijŇæ■ěéłd'åęĆäÿŃïijŽ

åĚĹçŤÍHomebrewåőL'èčĚæIJL'bugçŽĎDelveïijŽ

```
brew install go-delve/delve/delve
rm /usr/local/bin/dlv
```

åőL'èčĚæ■d'æIJL'éŮőécŸçŽĎDelveïijŇåĚűåőđåřśæŸŕäÿžäžĘèől'åőČåÿőæĹŚäžňåIJÍMacæIJžåŹÍäÿŁç■ certèŕĄäźęãĂĆåęĆæđIJä¡äèĞłåůśæĎ£æĎŘçźĄçŘŘçŽĎæL'ŃåŁĺåĹŻåžžèŕĄäźęïijŇäź§åŘŕäžěäÿ■çŤÍåőL'èčĚ a self-signed certificateãĂŚãĂĆ çňňäžŇæİąrmåŚ¡äžd'æŸŕäÿžäžĘåĹäéŹd'è£ŹäÿłæIJL'éŮőécŸçŽĎdlv binaryïijŇæĹŚäžňéIJĂèęĄäżŐæžŘçăĄçijŰèŕŚåĞžäÿĂäÿłæ■čçąőçŽĎçL'ĹæIJňïijŇåźűäÿŤåĹl'çŤÍHomebewä äÿŃè¡¡æžŘäžčçăĄ

```
mkdir -p /Users/xxx/go-delve/src/github.com/derekparker
cd /Users/xxx/go-delve/src/github.com/derekparker
git clone https://github.com/derekparker/delve.git
```

åĹŻåžžäÿĂäÿłäÿť æŮűæŰĞäžűåď źïijŇäžŐgithubäÿŃè¡¡äžčçăĄąãĂĆæşĺæĎŔæŰĞäžűåď źäÿ■æăĞæşĺçžc not foundãĂĆåĚűåőČéČĺåĹĘèŕůæăźæ■őèĞłåůśæIJžåŹĺçŐŕáćČèő¿ç¡őãĂĆ

çijŰèŕŚ

```
export GOPATH=/Users/xxx/go-delve
cd /Users/xxx/go-delve/src/github.com/derekparker/delve
make install
```

åžŤèŕěäijŽåĞžçŐřåęĆäÿŃæŔŘçď žïijŇèąĺæŸŐçijŰèŕŚæĹŘåŁ§ïijŽ

```
scripts/gencert.sh || (echo "An error occurred when generating and␣
↪installing a new certicate"; exit 1)
go install -ldflags="-s" github.com/derekparker/delve/cmd/dlv
codesign -s "dlv-cert"  /Users/xxx/go-delve/bin/dlv
```

çĎűåŘŐcp /Users/liuwb/go-delve/bin/dlv/usr/local/bin/ïijŇæŁŁçijŰèŕŚåě¡çŽĎdlvæŃůèt'İè£Ż/usr/local/b debuggerãĂĆè¿ŞåĚěåŚ¡äžd'dlv versionïijŇåęĆæđIJèČ¡æ■čåÿÿè£ŘèąŇïijŇæŸ¿çd'žçL'ĹæIJňåŕůïijŇèŕt'æŸŐ

åIJÍ Ubuntu äÿŁåőL'èčĚDelve

åŕźäžŐUbuntuçşžçž§ïijŇåŕŕäžěçŻt'æŐěä¡£çŤĺäÿŃéİćçŽĎæŇĞäžd'åőL'èčĚDelveïijŽ

```
go get -u github.com/derekparker/delve/cmd/dlv
```

äÿŃè¡¡Go-NebulasåůěḉŃäžčçăĄ

```
mkdir /Users/xxx/workspace/blockchain/src/github.com/nebulasio/
cd /Users/xxx/workspace/blockchain/src/github.com/nebulasio/
git clone https://github.com/nebulasio/go-nebulas.git
```

åĹŻåžžäÿĂäÿłäÿť æŮűæŰĞäžűåď źïijŇäžŐgithubäÿŃè¡¡äžčçăĄąãĂĆæşĺæĎŔæŰĞäžűåď źäÿ■æăĞæşĺçžc

DelveåŚ¡äžd'èąŇèřČèŕŢ åęĆæđIJä¡ääžěåL'■çŤĺgdbèřČèŕŢè£ĞCçĺŃåžŔïijŇåŕźdlvåŚ¡äžd'èąŇèřČèŕŢçŽ è£ŹéĞŇåŔłäžŃçż■debugéČĺåĹĘãĂĆ

è¿ŞåĚěåęĆäÿŃåŚ¡äžd'è£ŻåĚèèřČèŕŢ

```
export GOPATH=/Users/xxx/workspace/blockchain/
cd /Users/xxx/workspace/blockchain/
dlv debug github.com/nebulasio/go-nebulas/cmd/neb -- --config /
↪Users/xxx/workspace/blockchain/src/github.com/nebulasio/go-
↪nebulas/conf/default/config.conf
```

è£ŘèąŇæŮăèŕŕçŽĎèŕİïijŇäijŽè£ŻåĚědebug sessionïijŽ

```
Type 'help' for list of commands.
(dlv)
```

æĹŚäžňæL'ŞçőŮåIJĺnebçŽĎåĞ¡æŢřåĚěåŔčèő¿ç¡őæŰ■çĆźïijŇè¿ŞåĚěåŚ¡äžd'

```
(dlv) break main.neb
Breakpoint 1 set at 0x4ba6798 for main.neb() ./src/github.com/
↪nebulasio/go-nebulas/cmd/neb/main.go:80
(dlv)
```

dlvèřČèŕŢåŹĺæŔŘçd'žäžčçăĄåřĘåIJĺcmd/neb/main.goçŽĎèąŇåŔů80èąŇåĄIJä¡ŔïijŇæşĺæĎŔè£ŹæŮűne

```
(dlv) continue
> main.neb() ./src/github.com/nebulasio/go-nebulas/cmd/neb/main.
↪go:80 (hits goroutine(1):1 total:1) (PC: 0x4ba6798)
    75:         sort.Sort(cli.CommandsByName(app.Commands))
    76:
    77:         app.Run(os.Args)
    78:     }
    79:
=>  80:     func neb(ctx *cli.Context) error {
    81:         n, err := makeNeb(ctx)
    82:         if err != nil {
    83:             return err
    84:         }
    85:
```

æ§ěçIJŃåŔŸéĞŔïijŇåŔŕçŤĺprintåŚ¡äžd'ïijŽ

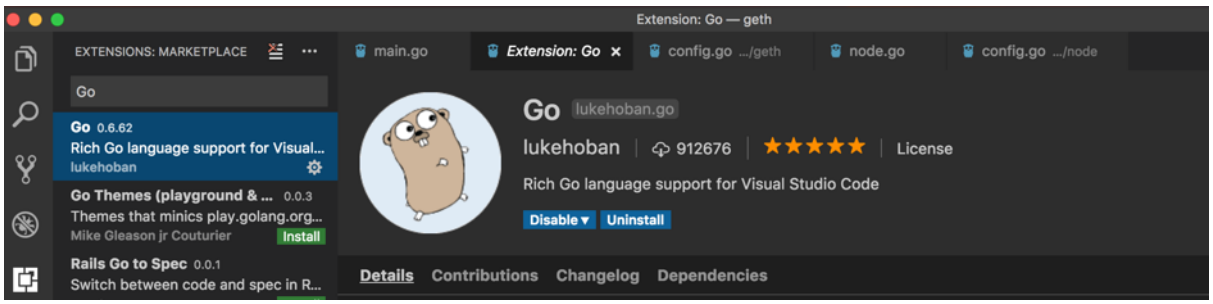
```
(dlv) print ctx
*github.com/nebulasio/go-nebulas/vendor/github.com/urfave/cli.
↪Context {
    App: *github.com/nebulasio/go-nebulas/vendor/github.com/urfave/
↪cli.App {
        Name: "neb",
        HelpName: "debug",
        Usage: "the go-nebulas command line interface",
        UsageText: "",
        ArgsUsage: "",
        Version: ", branch , commit ",
        Description: "",
        Commands: []github.com/nebulasio/go-nebulas/vendor/github.
↪com/urfave/cli.Command len: 11, cap: 18, [
            (*github.com/nebulasio/go-nebulas/vendor/github.com/
↪urfave/cli.Command)(0xc4201f4000),
            (*github.com/nebulasio/go-nebulas/vendor/github.com/
↪urfave/cli.Command)(0xc4201f4128),
            (*github.com/nebulasio/go-nebulas/vendor/github.com/
↪urfave/cli.Command)(0xc4201f4250),
            (*github.com/nebulasio/go-nebulas/vendor/github.com/
↪urfave/cli.Command)(0xc4201f4378),
            (*github.com/nebulasio/go-nebulas/vendor/github.com/
↪urfave/cli.Command)(0xc4201f44a0),
```

æŻt'åd'ŽæŁĂæIJŕèţĎæŰŹïijŇèŕůåŔĆèĂČ https://github.com/derekparker/

delve/tree/master/Documentation/cli https://blog.gopheracademy.com/advent-2015/debugging-with-delve/ http://hustcat.github.io/getting-started-with-delve/

Visual Studio CodeèřČèŕŢ

Visual Studio CodeæŸŕå¿őè¡ŕåĚňåŔŷåŔŠåŷČçŽĎèůĺåźşåŔřäžčçăAçijŰè¿ŠåůěåĚůïijŇäŷŃè¡¡åIJřåİĂïijŽ //code.visualstudio.com/Download VS CodeéIJĂèęĄåőL'èčĚGoæŔŠäžű



æL'ŞåijĂæŰĞäžűåď'ź/Users/xxx/workspace/blockchain/src/github.com/nebulasio/go-nebulas/ïijŇåIJÍ.vscodeæŰĞäžűåď'źäŷŃåĹŻåžžäŷď'äŷłæŰĞäžűsettings.jsonåŠŇlaunch.jsonãĂĆ settings.jsonæŰĞäžűåĘĚåőźïijŽ

```
// Place your settings in this file to overwrite default and user␣
↪settings.
{
    "go.gopath": "/Users/xxx/workspace/blockchain/",
    "go.formatOnSave": true,
    "go.gocodeAutoBuild": false,
    "go.toolsGopath": "/Users/xxx/workspace/gotools",
    "explorer.openEditors.visible": 0,
}
```

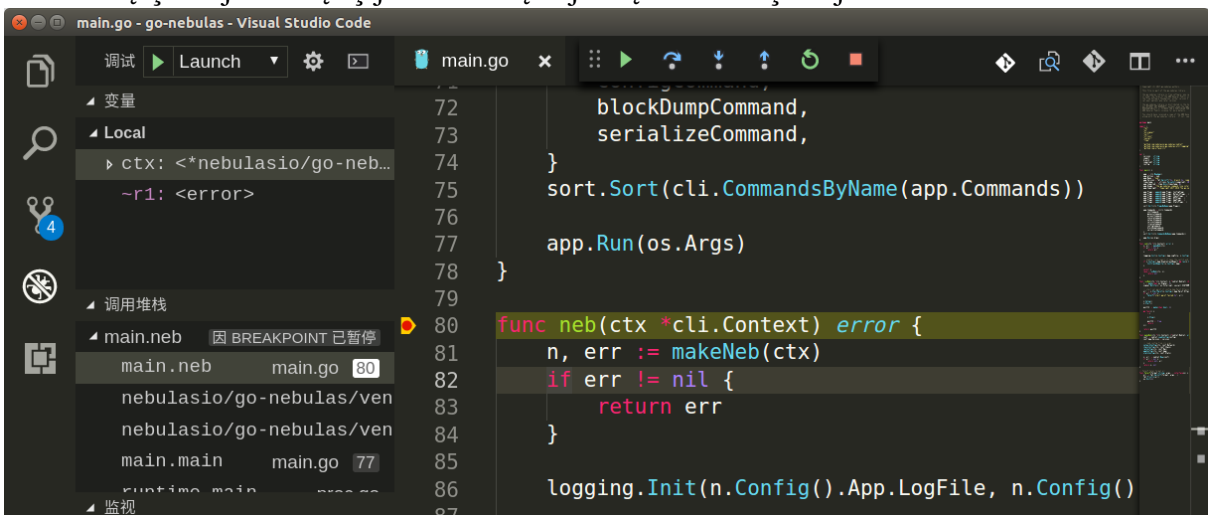
go.toolsGopathæŸŕanalysis toolsåőL'èčĚçŽĎåIJřåİĂïijŇåŔŕäžěæŇĞåőŽäŷžäżżä¡ŢçŻőå¡ŢïijŇè£ŹäžŻan toolsåŔŕäžěä¿ŻåĚűåőČworkspaceåĚšäžńãĂĆ

launch.jsonæŰĞäžűåĘĚåőźïijŽ

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Launch",
            "type": "go",
            "request": "launch",
            "mode": "debug",
            "program": "${workspaceRoot}/cmd/neb",
            "env": {
                "GOPATH": "/Users/xxx/workspace/blockchain/"
            },
            "args": [
                "--config",
```

```
                "/Users/xxx/workspace/blockchain/src/github.com/
↪nebulasio/go-nebulas/conf/default/config.conf"
            ],
            "showLog": true
        }
    ]
}
```

åIJĺcmd/neb/main.goïijŇnebåĞ¡æŢřäÿ■èő¿ç¡őæŰ■çĆźïijŇF5è£ŘèąŇïijŇGo-NebulaséąźçŽőäijŽè£ŻèąŇçijŰèŕŚè£ŘèąŇïijŇåĄIJåIJĺæŰ■çĆźïijŽ



çĎűåŘŐïijŇåřšåŔŕäžěåijĂå£ČçŽĎåŘŕåŁÍNebulasäžčçăĄèřČèŕŢäźŃæŮĚïijĄ

debuging-with-gdb

OverView

Last week we found a lot of âĂIJFailed to update latest irreversible block.âĂİ in neb log with Leon. The reference code (nebulasio/go-nebulas/core/blockchain.go updateLatestIrreversibleBlock) ïijŇ in the code we found the cur variable is not equal to the tail variable , why? to find the cause, we try to use tool to dynamically display variable information and facilitate single-step debugging.

Goroutines

In c++ program we often use gbd to debug, so we think why not to use gdb to debug golang program . First we try to look up the BlockChain loop goroutine state and print the variables .

In c++ we all use `info threads` and thread x to show thread info but in the golang program ïijŇwe should use `info goroutines` and `goroutine xx bt` to displays the current list of running goroutines.

(gdb) `info goroutines` Undefined info command: “goroutines“. Try “help info“. (gdb) `source` /usr/local/go/src/runtime/runtime-gdb.py Loading Go Runtime support. (gdb)

`info goroutines`

```
1 waiting  runtime.gopark
2 waiting  runtime.gopark
3 waiting  runtime.gopark
4 waiting  runtime.gopark
5 syscall  runtime.notetsleepg
6 syscall  runtime.notetsleepg
7 waiting  runtime.gopark
... ...
```

(gdb) goroutine 84 bt

```
 #0 runtime.gopark (unlockf={void (struct runtime.g , void , bool␣
↪*)} 0xc420c57c80, lock=0x0, reason="select", traceEv=24 '\030',␣
↪traceskip=1) at /data/packages/go/src/runtime/proc.go:288
 #1 0x0000000000440fd9 in runtime.selectgo (sel=0xc420c57f48, ~
↪r1=842353656960) at /data/packages/go/src/runtime/select.go:395
 #2 0x0000000000ad2d73 in github.com/nebulasio/go-nebulas/core.
↪(*BlockChain).loop (bc=0xc4202c6320)at /neb/golang/src/github.com/
↪nebulasio/go-nebulas/core/blockchain.go:184
 #3 0x0000000000460421 in runtime.goexit () at /data/packages/go/
↪src/runtime/asm_amd64.s:2337
 #4 .....
```

But neb has too many goroutines, we donâĂŹt kown which one , we give up

BreakPoints

Second we try to set break point to debug

(gdb) `b blockchain.go:381`

Breakpoint 2 at 0xad4373: file /neb/golang/src/github.com/nebulasio/go-nebulas/core/blockchain.go, line 381.

(gdb) `b core/blockchain.go:390`

Breakpoint 3 at 0xad44c6: file /neb/golang/src/github.com/nebulasio/go-nebulas/core/blockchain.go, line 390.

(gdb) `info breakpoints` // show all breakpoints

(gdb) `d 2` //delete No 2 breakpoint

Now let the neb continue its execution until the next breakpoint, enter the c command: (gdb) `c` Continuing

```
Thread 6 "neb" hit Breakpoint 2, github.com/nebulasio/go-nebulas/
↪core.(*BlockChain).updateLatestIrreversibleBlock (bc=0xc4202c6320,
↪ tail=0xc4244198c0)
at /neb/golang/src/github.com/nebulasio/go-nebulas/core/blockchain.
↪go:382
```

```
382         miners := make(map[string
```

now we can use p(print) to print variables value

```
(gdb) `p cur`
$2 = (struct github.com/nebulasio/go-nebulas/core.Block *)␣
↪0xc420716f90
(gdb) `p cur.height`
$3 = 0
(gdb) `p bc`
$4 = (struct github.com/nebulasio/go-nebulas/core.BlockChain *)␣
↪0xc4202c6320
(gdb) `p bc.latestIrreversibleBlock`
$5 = (struct github.com/nebulasio/go-nebulas/core.Block *)␣
↪0xc4240bbb00
(gdb) `p bc.latestIrreversibleBlock.height`
$6 = 51743
(gdb) `p tail`
$7 = (struct github.com/nebulasio/go-nebulas/core.Block *)␣
↪0xc4244198c0
(gdb) `p tail.height`
$8 = 51749
```

now we can use `info goroutines` again, to find current goroutine. info goroutines with the * indicating the current execution, so we find the current goroutine nunmber quickly.

the next breakpoint we can use `c` command , so we found the cur and lib is not equal, because of length of the miners is less than ConsensusSizeïijŇ In the loop the cur change to the parent block .

Other

When compiling Go programs, the following points require particular attention:

- Using -ldflags "-s" will prevent the standard debugging information from being printed
- Using -gcflags "-N-l" will prevent Go from performing some of its automated optimizations -optimizations of aggregate variables, functions, etc. These optimizations can make it very difficult for GDB to do its job, so it's best to disable them at compile time using these flags.

References

- Debugging with GDB
- GDBèřČèŕŢGOçÍŃåžŔ

neb-dont-generate-coredump-file

OverView

During Testing, neb may be crash, and we want to get the coredump file which could help us to find the reason. However, neb don‘t generate coredump file by default. We can find the crash log in /var/log/apport.log when a crash occurred:

```
"called for pid 10110, signal 11, core limit 0, dump mode 1 "
```

The coredump file is very very important, it can serve as useful debugging aids in several situations, and help us to debug quickly. Therefore we should make neb to generate coredump file.

Set the core file size

We can use `ulimit -a` command to show core file size. If it‘s size is zero, which means coredump file is disabled, then we should set a value for core file size. for temporarily change we can use `ulimit -c unlimited` , and for permanently change we can edit `/etc/security/limits.conf` file, it will take effect after reboot or command `sysctl -p`.

```
<domain>    <type>    <item>         <value>
 * soft    core             unlimited
```

But these ways are‘t work, neb still can‘t generate coredump file and `cat /proc/$pid/limits` always “Max core file size 0“

Why? Why? Why? It doesn‘t Work

1. If the setting is wrong? Just try a c++ programe build, run it and we can find that it can generate coredump.
2. Neb is started by supervisord, is it caused by supervisordïij§
3. Try to start neb without supervisord, then the neb coredump is generated!
4. Yes, the reason is supervisord, then we can google “supervisord+coredump“ to solve it.

Solution

Supervisord only set RLIMIT_NOFILE, RLIMIT_NOPROC by set_rlimits , others are seted default 0 1. modify supervisord code options.py in 1293 line

```
vim /usr/lib/python2.6/site-packages/supervisor/options.py
```

```
soft, hard = resource.getrlimit(resource.RLIMIT_CORE)
resource.setrlimit(resource.RLIMIT_CORE, (-1, hard))
```

1. restart supervisord and it works .

Other seetings

You can also change the name and path of coredump file by changing file `/proc/sys/kernel/core_pattern`:

```
echo "/neb/app/core-%e-%p-%t" > /proc/sys/kernel/core_pattern

%p: pid
%: '%' is dropped
%%: output one '%'
%u: uid
%g: gid
%s: signal number
%t: UNIX time of dump
%h: hostname
%e: executable filename
%: both are dropped
```

References

- supervisord coredump
- core_pattern

Tutoriais

Nebulas 101 - 01 CompilaÃğÃčo e InstalaÃğÃčo de Nebulas

A versÃčo actual da Mainnet da Nebulas Ãl' 2.0, denominada Nebulas Nova.

A Nebulas Nova visa descobrir o valor dos dados da blockchain, e representa o futuro da colaboraÃğÃčo.

Veja a nossa introduÃğÃčo no YouTube para uma explicaÃğÃčo mais detalhada..

Pode descarregar o cÃşdigo fonte da Nebulas e compilar a blockchain privada localmente.

Para aprender sobre Nebulas, por favor leia o White Paper NÃčo-TÃl'cnico de Nebulas.

Para aprender sobre a tÃl'cnologia, por favor leia o White Paper TÃl'cnico e cÃşdigo no github de Nebulas.

> Nebulas apenas pode correr em Mac e Linux, de momento. A versÃčo Windows irÃą ser lanÃğada brevevemente.

Ambiente de desenvolvimento para Golang

Para jÃą, Nebulas foi implementada em Golang e C++.

Mac OSX

Homebrew Ãl' recomendado para instalar Golang em Mac.

```
# install
brew install go

# environment variables
export GOPATH=/caminho/para/areadetrabalho
```

Nota: GOPATH Ãl' o directÃşrio de trabalho que pode ser escolhido por si. Depois de configurar o GOPATH, os seus projectos Go terÃčo de ser colocados no directÃşrio GOPATH.

Linux

```
# download
wget https://dl.google.com/go/go1.12.linux-amd64.tar.gz

# extraÃğÃčo
tar -C /usr/local -xzf go1.12.linux-amd64.tar.gz

# variÃąveis do ambiente de trabalho
export PATH=$PATH:/usr/local/go/bin
export GOPATH=/caminho/para/areadetrabalho
```

CompilaÃğÃčo Nebulas

Download

Clone o cÃşdigo fonte com os seguintes comandos:

```
# entra na Ãąrea de trabalho
cd /caminho/para/areadetrabalho

# descarrega
git clone https://github.com/nebulasio/go-nebulas.git

# entra no repositÃşrio
cd go-nebulas
```

```
# o branch/ramo principal Ãl' o mais estÃąvel
git checkout master
```

Compile Neb

Prepare o seu ambiente de execuÃğÃčo:

```
cd /caminho/para/areadetrabalho
source setup.sh
```

Compile NEB. Pode agora criar o executÃąvel da Nebulas:

```
cd /caminho/para/areadetrabalho
make build
```

Uma vez que a compilaÃğÃčo tiver acabado irÃą encontrar um ficheiro `neb` executÃąvel no directÃşrio raiz.

```
➜ go-nebulas git:(master) ✗ make build
cd cmd/neb; CGO_CFLAGS="-I/Users/congming/go/src/github.com/nebulasio/go-nebulas/nbre/lib/include -g -O2" CGO_LDFLAGS="-L/Users/congming/go/src/github.com/nebulasio/go-nebulas/native-lib -lrocksdb -lstdc++ -lc++ -lgflags -lm -lz -lbz2 -lsnappy -llz4 -lzstd -g -O2" go build -ldflags "-X main.version=2.0 -X main.commit=9a6703fb1fa377ad0d0230f0f99d85e5d684144d -X main.branch=master -X main.compileAt=`date +%s`" -o ../../neb
```

make build

Execute NEB

Bloco GÃľnese

Antes de lanÃğar uma nova chain Nebulas, temos de definir a configuraÃğÃčo do bloco gÃľnese.

ConfiguraÃğÃčo do Bloco GÃľnese

```
# Ficheiro de texto Neb gÃl'nese. Esquema Ãl' definido em core/pb/
↪genesis.proto.

meta {
    # Identidade da chain
  chain_id: 100
}

consensus {
    dpos {
    # Dinastia inicial, incluÃ■ndo todos os mineradores iniciais
        dynasty: [
            [ miner address ],
            ...
        ]
```

```
    }
}

# PrÃl'-atribuiÃğÃčo dos tokens iniciais
token_distribution [
    {
        address: [ endereÃğo de atribuiÃğÃčo ]
        value: [ quantidade de atribuiÃğÃčo de tokens ]
    },
    ...
]
```

Um exemplo de uma genesis.conf encontra-se em `conf/default/genesis.conf`.

ConfiguraÃğÃčo

Antes de activar o nÃş neb, temos de definir a configuraÃğÃčo do mesmo.

ConfiguraÃğÃčo do NÃş Neb

```
# Ficheiro de configuraÃğÃčo Neb. Esquema Ãl' definido em neblet/pb/
↪config.proto:Config.

# ConfiguraÃğÃčo da rede
network {
    # Para o primeiro nÃş numa novo chain Nebulas, nÃčo Ãl' preciso␣
↪uma `seed` (semente).
    # Caso contrÃąrio, todo o nÃş precisa de nÃşs semente para␣
↪serem introduzidos na chain de Nebulas.
    # semente: ["/ip4/127.0.0.1/tcp/8680/ipfs/
↪QmP7HDFcYmJL12Ez4ZNVCKjKedfE7f48f1LAkUc3Whz4jP"]

    # Host de serviÃğo de rede P2p. Suporta mÃžltiplos Ips e portas.
    listen: ["0.0.0.0:8680"]

    # A chave privada Ãl' usada para gerar o ID do nÃş. Se nÃčo␣
↪usar chave privada, o nÃş vai gerar um novo ID de nÃş.
    # private_key: "conf/network/id_ed25519"
}

# ConfiguraÃğÃčo da Chain
chain {
    # ID da Chain de Rede
    chain_id: 100

    # Local de armazenamento da base de dados
    datadir: "data.db"
```

```
    # Local da keystore das contas
    keydir: "keydir"

    # ConfiguraÃğÃčo do bloco de gÃl'nese
    genesis: "conf/default/genesis.conf"

    # Algoritmo da assinatura
    signature_ciphers: ["ECC_SECP256K1"]

    # EndereÃğo do minerador
    miner: "n1SAQy3ix1pZj8MPzNeVqpAmu1nCVqb5w8c"

    # EndereÃğo Coinbase, todas as recompensas recebidas pelo␣
↪minerador acima serÃčo enviadas para este endereÃğo
    coinbase: "n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE"

    # A palavra passe para o ficheiro da keystore do minerador
    passphrase: "passphrase"
}

# ConfiguraÃğÃčo do API
rpc {
    # Porta do API GRPC
    rpc_listen: ["127.0.0.1:8684"]

    # Porta do API HTTP
    http_listen: ["127.0.0.1:8685"]

    # O mÃşdulo aberto
    http_module: ["api", "admin"]
}

# ConfiguraÃğÃčo do registo
app {
    # NÃ■vel de registo: [debug, info, warn, error, fatal]
    log_level: "info"

    # Local do registo
    log_file: "logs"

    # Abrir registo de falhas
    enable_crash_report: false
}

# ConfiguraÃğÃčo do NBRE
nbre {
    # directÃşrio raÃ■z do NBRE, onde as bibliotecas se encontram
    root_dir: "nbre"
    # path do directÃşrio dos registos do NBRE
    log_dir: "conf/nbre/logs"
```

```
    # directÃşrio da db do NBRE
    data_dir: "conf/nbre/nbre.db"
    # directÃşrio do executÃąvel do NBRE
    nbre_path: "nbre/bin/nbre"
    # EndereÃğo do administrador usado para submeter transacÃğÃţes␣
↪e autorizar contas especÃ■ficas
    # a fazer submissÃţes IR. Para mais detalhes, por favor␣
↪consulte os documentos do NBRE.
    admin_address: "n1S9RrRPC46T9byYBS868YuZgzqGuiPCY1m"
    # Altura quando o DIP entra em efeito
    start_height: 2307000
    # Socket IPC do NEB e NBRE
    ipc_listen: "127.0.0.1"
    ipc_port: 8688
}

# ConfiguraÃğÃčo de mÃl'tricas
stats {
    # Abrir nÃş de mÃl'tricas
    enable_metrics: false

    # ConfiguraÃğÃčo influxdb
    influxdb: {
        host: "http://localhost:8086"
        db: "nebulas"
        user: "admin"
        password: "admin"
    }
}
```

Muitos exemplos podem ser encontrados em `$GOPATH/src/github.com/nebulasio/go-nebulas/conf/`

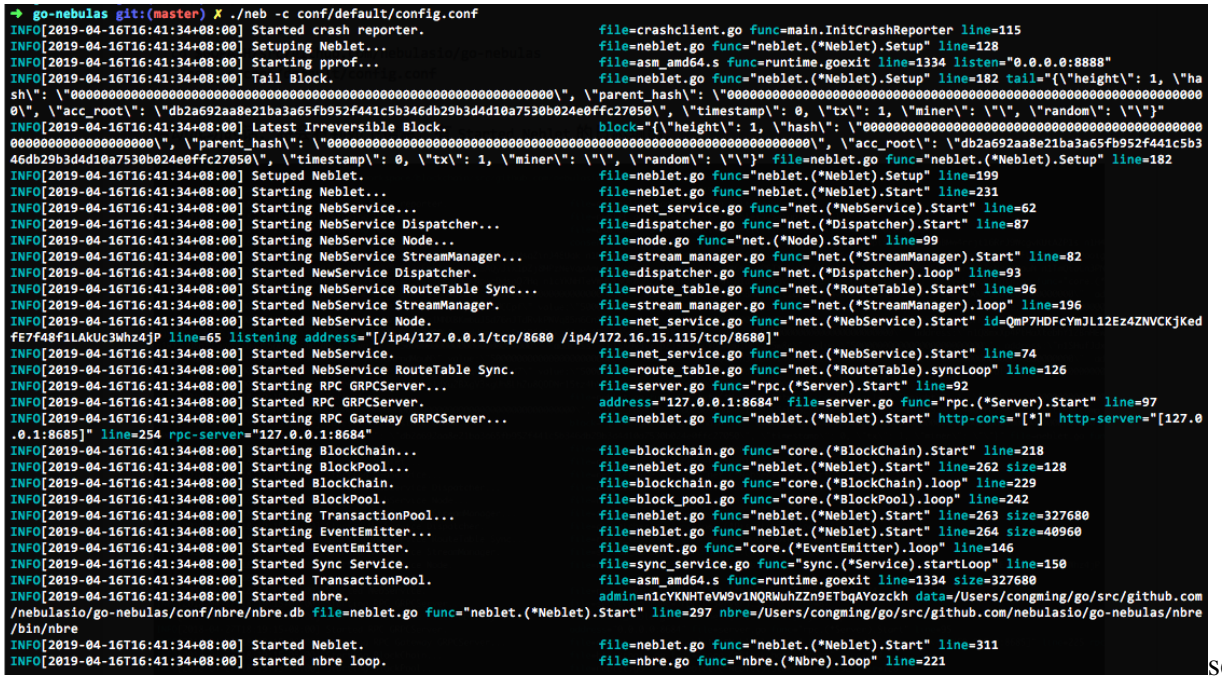
Operar nÃşs

A chain Nebulas que estÃą a operar de momento Ãl' privada e diferente da Testnet e Mainnet oficial.

Activa o teu primeiro nÃş Nebulas com os seguintes comandos:

```
cd $GOPATH/src/github.com/nebulasio/go-nebulas
./neb -c conf/default/config.conf
```

Depois de iniciar, o seguinte deverÃą aparecer no terminal:

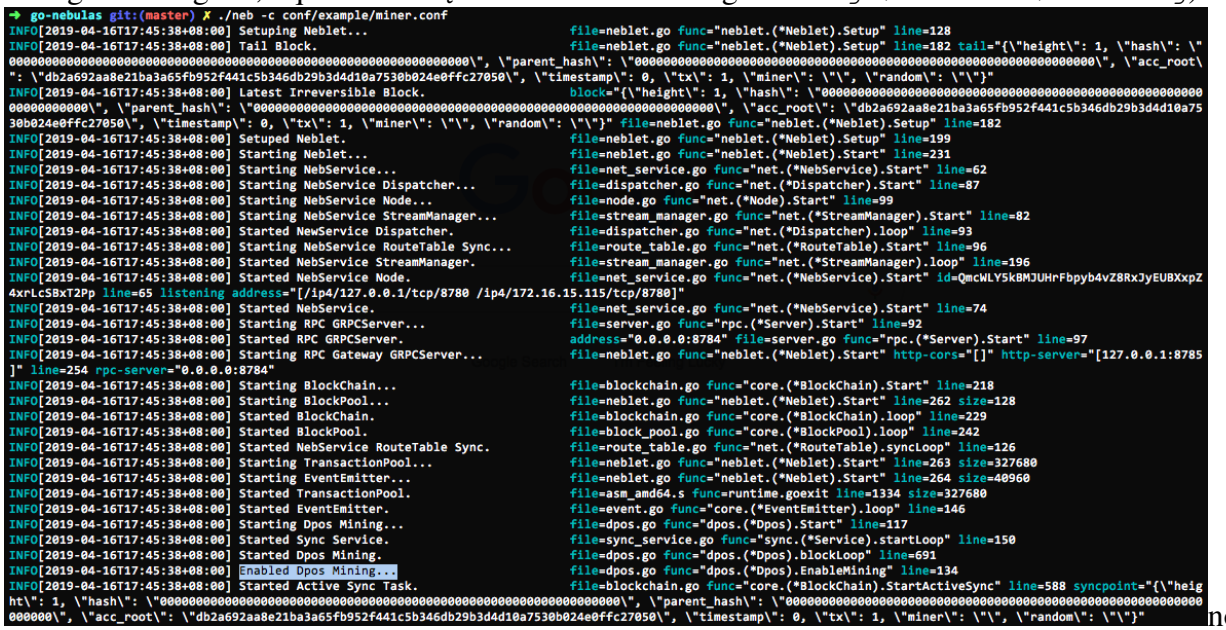


seed node start

Por padrÃčo, o nÃş que usa `conf/default/config.conf` nÃčo irÃą minerar novos blocos. Activa o teu primeiro nÃş de mineraÃğÃčo Nebulas com os prÃşximos comandos:

```
cd $GOPATH/src/github.com/nebulasio/go-nebulas
./neb -c conf/example/miner.conf
```

Depois do nÃş iniciar, se a conexÃčo com o nÃş semente suceder, irÃą ver o seguinte registo, que estarÃą no ficheiro de registo `logs/miner.1/neb.log`):



node start

> Nota: Pode iniciar vÃąrios nÃşs localmente. Por favor confirme que as portas nos ficheiros de configuraÃğÃčo dos nÃşs sÃčo diferentes para nÃčo haverem conflictos.

PrÃşximo passo: Tutorial 2

Envio de TransaÃğÃţes com Nebulas

Nebulas 101 - 02 Envio de TransacÃğÃţes na Nebulas

Tutorial Youtube

Nesta porÃğÃčo do tutorial vamos continuar o que estÃąvamos a fazer no Tutorial de InstalaÃğÃčo 1.

Nebulas dispÃţe de trÃłs mÃl'todos para enviar transaÃğÃţes:

1. Assina & Envia
2. Envia com senha
3. Desbloqueia & Envia

Aqui estÃą uma introduÃğÃčo ao envio de uma transaÃğÃčo de Nebulas atravÃl's dos trÃłs mÃl'todos acima descritos, e a verificaÃğÃčo do sucesso da transaÃğÃčo.

PreparaÃğÃčo de Contas

Com Nebulas, cada endereÃğo representa uma conta Ãžnica.

Prepara duas contas: um endereÃğo para enviar tokens (o endereÃğo de envio/remetente, chamado âĂIJfromâĂİ), e o endereÃğo para receber os tokens (o endereÃğo de recepÃğÃčo/destinatÃąrio, chamado âĂIJtoâĂİ).

O Remetente

Aqui vamos utilizar uma conta Coinbase no `conf/example/miner.conf`, que usa `n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE` como remetente. Sendo a conta Coinbase do mineiro, irÃą receber uns tokens como recompensa de mineraÃğÃčo. Mais tarde podemos enviar esses tokens para outra conta.

O DestinatÃąrio

Crie uma nova wallet/carteira para receber os tokens.

```
$ ./neb account new
A sua conta estÃą bloqueada com uma senha, Por favor defina uma␣
 ↪senha. NÃčo se esqueÃğa desta senha.
Passphrase:
Repeat passphrase:
Address: n1SQe5d1NKHYFMKtJ5sNHPsSPVavGzW71Wy
```

Quando executar este comando a carteira/wallet terÃą um endereÃğo diferente, com `n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE`. Por favor use o endereÃğo gerado como o destinatÃąrio.

O ficheiro keystore da nova carteira ficarÃą em `$GOPATH/src/github.com/nebulasio/go-nebulas/keydir/`

ActivaÃğÃčo dos NÃşs

ActivaÃğÃčo do NÃş Semente

Primeiro, active o nÃş semente como o primeiro nÃş da blockchain local privada.

```
./neb -c conf/default/config.conf
```

Active o NÃş Mineiro

Em segundo lugar, inicÃ■e o nÃş mineiro ligado ao nÃş semente. Este nÃş irÃą gerar novos blocos na chain local privada.

```
./neb -c conf/example/miner.conf
```

Quanto tempo atÃl' um novo bloco ser cunhado?

Em Nebulas, DpoS foi escolhido como o algoritmo para o mecanismo temporÃąrio de consenso antes do Proof-of-Devotion (PoD, descrita no White Paper TÃl'cnico) estar pronto. Neste algoritmo de consenso, cada mineiro vai cunhar um novo bloco a cada 15 segundos.

Em contexto corrente, temos de esperar 315(=15*21) segundos atÃl' cunhar um novo bloco pois apenas hÃą um mineiro a trabalhar agora, entre os 21 mineiros definidos em `conf/default/genesis.conf`.

Logo que um novo bloco tenha sido cunhado pelo mineiro, a recompensa da mineraÃğÃčo serÃą adicionada ao endereÃğo da carteira Coinbase delineado em `conf/example/miner.conf`, que Ãl' `n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE`.

InteraÃğÃčo com os NÃşs

Nebulas fornece os criadores com API HTTP, API gRPC, e CLI para interagir com os nÃşs em funcionamento. Aqui, vamos partilhar trÃłs mÃl'todos para enviar uma transaÃğÃčo, com o API HTTP (API Module | Admin Module).

O ouvinte de HTTP da Nebulas encontra-se definido na configuraÃğÃčo do nÃş. A porta padrÃčo do nosso nÃş semente Ãl' `8685`.

Para comeÃğar, verifique o balanÃğo do remetente antes de enviar a transaÃğÃčo.

VerificaÃğÃčo do Estado da Conta

Adquira o estado da conta do remetente `n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE` com `/v1/user/accountstate` no API Module, usando `curl`.

```
> curl -i -H Accept:application/json -X POST http://localhost:8685/
↪v1/user/accountstate -d '{"address":
↪"n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE"}'

{
    "result": {
        "balance": "50000000000000000000000000",
        "nonce": "0",
        "type": 87
        "height": "1"
        "pending": "0"
    }
}
```

Nota O tipo Ãl' usado para verificar se esta conta Ãl' uma conta de smart contract. `88` representa uma conta do tipo smart contract, e `87` uma conta non-contract. Height Ãl' usado para indicar a altura actual da blockchain quando o API Ãl' chamado. Pending Ãl' usado para determinar o nÃžmero de transacÃğÃţes pendentes se encontram na sua pool de transacÃğÃţes.

Como pode ver, o destinatÃąrio foi recompensado com alguns tokens por ter minerado novos blocos.

EntÃčo vamos verificar o estado da conta do destinatÃąrio/recipiente.

```
> curl -i -H Accept:application/json -X POST http://localhost:8685/
↪v1/user/accountstate -d '{"address":"o_seu_endereÃğo"}'

{
    "result": {
        "balance": "0",
        "nonce": "0",
        "type": 87
        "height": "1"
        "pending": "0"
    }
}
```

A nova conta nÃčo tem tokens, como esperado.

Envio de TransaÃğÃčo

Agora vamos enviar uma transaÃğÃčo usando trÃłs mÃl'todos para transferir alguns tokens do remetente para o destinatÃąrio!

Assina & Envia

Desta forma, pode assinar uma transaÃğÃčo num ambiente sem rede (offline), e depois submetÃł-la a um nÃş que esteja online. Este Ãl' o mÃl'todo mais seguro para submeter uma transaÃğÃčo sem expor a sua chava privada Ãă Internet.

Primeiro, assine a transaÃğÃčo para obter os dados brutos.

```
> curl -i -H 'Content-Type: application/json' -X POST http://
 ↪localhost:8685/v1/admin/sign -d '{"transaction":{"from":
 ↪"n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE","to":
 ↪"n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5", "value":
 ↪"1000000000000000000","nonce":1,"gasPrice":"20000000000","gasLimit
 ↪":"2000000"}, "passphrase":"passphrase"}'

{"result":{"data":"CiAbjMP5dyVsTWILfXL1MbwZ8Q6xOgX/
 ↪JKinks1dpToSdxIaGVcH+WT/
 ↪SVMkY18ix7SG4F1+Z8evXJoA35caGhlXbip8PupTNxwV4SRM87r798jXWADXpWngIhAAAAAAAAAAAA
 ↪"}}
```

> **Nota** Nonce Ãl' um atributo deveras importante numa transaÃğÃčo. Foi engendrado para prevenir ataques de replay. Para uma conta especifica, apenas apÃşs a transaÃğÃčo com nonce N for aceite Ãl' que a transaÃğÃčo com nonce N+1 serÃą processada. Logo, temos de verificar o ultimo nonce da conta na chain antes de preparar uma nova transaÃğÃčo.

Depois, envia os dados brutos para um nÃş Nebulas online.

```
> curl -i -H 'Content-Type: application/json' -X POST http://
 ↪localhost:8685/v1/user/rawtransaction -d '{"data":
 ↪"CiAbjMP5dyVsTWILfXL1MbwZ8Q6xOgX/JKinks1dpToSdxIaGVcH+WT/
 ↪SVMkY18ix7SG4F1+Z8evXJoA35caGhlXbip8PupTNxwV4SRM87r798jXWADXpWngIhAAAAAAAAAAAA
 ↪"}'

{"result":{"txhash":
 ↪"1b8cc3f977256c4d620b7d72f531bc19f10eb13a05ff24a8a792cd5da53a1277
 ↪","contract_address":""}}Ãć?Å¡
```

Envio com Senha

Se confia no nÃş Nebulas tanto que lhe pode delegar os seus ficheiros keystore, o segundo mÃl'todo Ãl' uma boa opÃğÃčo para si.

Primeiro, carregue os seus ficheiros keystore para os directÃşrios da chave no seu nÃş Nebulas de confianÃğa.

Depois, envie a transaÃğÃčo com a sua senha.

```
> curl -i -H 'Content-Type: application/json' -X POST http://
 ↪localhost:8685/v1/admin/transactionWithPassphrase -d '{
 ↪"transaction":{"from":"n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE","to":
 ↪"n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5", "value":
 ↪"1000000000000000000","nonce":2,"gasPrice":"20000000000","gasLimit
 ↪":"2000000"},"passphrase":"passphrase"}'

{"result":{"txhash":
 ↪"3cdd38a66c8f399e2f28134e0eb556b292e19d48439f6afde384ca9b60c27010
 ↪","contract_address":""}}
```

Nota Por ter enviado uma transaÃğÃčo com nonce 1 da conta `n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE`, a nova transaÃğÃčo com o mesmo `from` (remetente) deve ser incrementada por 1, para 2.

Desbloquear & Enviar

Este Ãl' o mÃl'todo mais perigoso. ProvÃąvelmente nÃčo o deve utilizar a nÃčo ser que tenha confianÃğa absoluta no nÃş Nebulas recipiente.

Primeiro, carregue os seus ficheiros keystore para os directÃşrios da chave no seu nÃş Nebulas de confianÃğa.

Depois desbloqueie as suas contas com a sua senha para uma duraÃğÃčo especÃ■fica no nÃş. A unidade de duraÃğÃčo Ãl' nano segundos (300000000000=300s).

```
> curl -i -H 'Content-Type: application/json' -X POST http://
 ↪localhost:8685/v1/admin/account/unlock -d '{"address":
 ↪"n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE","passphrase":"passphrase",
 ↪"duration":"300000000000"}'

{"result":{"result":true}}
```

Depois de desbloquear a conta, qualquer um serÃą capaz de enviar qualquer transaÃğÃčo directamente dentro da duraÃğÃčo definida nesse nÃş sem a sua autorizaÃğÃčo.

```
> curl -i -H 'Content-Type: application/json' -X POST http://
 ↪localhost:8685/v1/admin/transaction -d '{"from":
 ↪"n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE","to":
 ↪"n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5", "value":
 ↪"1000000000000000000","nonce":3,"gasPrice":"20000000000","gasLimit
 ↪":"2000000"}'

{"result":{"txhash":
 ↪"8d69dea784f0edfb2ee678c464d99e155bca04b3d7e6cdba6c5c189f731110cf
 ↪","contract_address":""}}Ãć?Å¡
```

Recibo da TransaÃğÃčo

Vai obter uma `txhash` nos trÃłs mÃl'todos apÃşs efectuar a transaÃğÃčo com sucesso. O valor do `txhash` pode ser usado para verificar o estado da transaÃğÃčo.

```
> curl -i -H Accept:application/json -X POST http://localhost:8685/
↪v1/user/getTransactionReceipt -d '{"hash":
↪"8d69dea784f0edfb2ee678c464d99e155bca04b3d7e6cdba6c5c189f731110cf
↪"}'

{"result":{"hash":
↪"8d69dea784f0edfb2ee678c464d99e155bca04b3d7e6cdba6c5c189f731110cf
↪","chainId":100,"from":"n1FF1nz6tarkDVwWQkMnnwFPuPKUaQTdptE","to":
↪"n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5","value":"1000000000000000000
↪","nonce":"3","timestamp":"1524667888","type":"binary","data
↪":null,"gas_price":"20000000000","gas_limit":"2000000","contract_
↪address":"","status":1,"gas_used":"20000"}}Ãć?Å¡
```

O `status` pode tomar os valores de 0, 1, ou 2.

- **0: Falha.** Significa que a transaÃğÃčo foi submetida Ãă chain mas a sua execuÃğÃčo falhou.
- **1: Sucesso.** Significa que a transaÃğÃčo foi submetida Ãă chain e a sua execuÃğÃčo foi um sucesso.
- **2: Pendente.** Significa que a transaÃğÃčo ainda nÃčo foi empacotada num bloco.

Dupla VerificaÃğÃčo

Vamos verificar novamente o balanÃğo do destinatÃąrio/recipiente.

```
> curl -i -H Accept:application/json -X POST http://localhost:8685/
↪v1/user/accountstate -d '{"address":
↪"n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5"}'

{"result":{"balance":"3000000000000000000","nonce":"0","type":87,
↪"height":"10","pending":"0"}}
```

Aqui deve ver o balanÃğo, que Ãl' o total de todas as transferÃłncias efectuadas com sucesso que executou.

PrÃşximo passo: Tutorial 3

Programa e executa um smart contract com JavaScript

Nebulas 101 - 03 Programar e executar um smart contract

Tutorial YouTube

AtravÃl's deste tutorial irÃą aprender como escrever, implementar, e executar smart contracts em Nebulas.

PreparaÃğÃčo

Antes de preencher o smart contract, primeiro reveja os conteÃždos prÃl'viamente leccionados:

1. InstalaÃğÃčo, compilaÃğÃčo e inicializaÃğÃčo de uma aplicaÃğÃčo neb
2. CriaÃğÃčo de um endereÃğo da carteira, configuraÃğÃčo da coinbase, a inicializaÃğÃčo de mineraÃğÃčo
3. Inquirir informaÃğÃčo ao nÃş de neb, endereÃğo da carteira, e balanÃğo
4. Envio de transaÃğÃčo e verificaÃğÃčo do sucesso da mesma

Se tiver dÃžvidas dos conteÃždos acima deve voltar atrÃąs para rever os capÃ■tulos relevantes. Vamos fazer isto. Vamos aprender a usar smart contracts atravÃl's das seguintes etapas:

1. Escrever o smart contract
2. Implementar o smart contract
3. Chamar o smart contract, e verificar os resultados da sua execuÃğÃčo

Programar um Smart Contract

Tal como o Ethereum, Nebulas implementa uma mÃąquina virtual (NVM) para executar smart contracts. A implementaÃğÃčo do NVM usa o interpretador JavaScript V8, portanto de momento podemos escrever smart contracts usando JavaScript e TypeScript.

Escreve uma especificaÃğÃčo breve de um smart contract:

1. O cÃşdigo do smart contract tem de ser um objecto Prototype;
2. O cÃşdigo do smart contract tem de ter um mÃl'todo init(), que apenas serÃą executado uma vez durante a implementaÃğÃčo;
3. Os mÃl'todos privados do smart contract tem de ser prefixados com _ , e nÃčo podem ser chamados directamente for a do contracto;

Abaixo usamos JavaScript para escrever o primeiro smart contract: cofre de banco. Este smart contract tem de cumprir as seguintes funÃğÃţes:

1. Utilizadores pode depositar dinheiro neste cofre bancÃąrio.
2. Utilizadores podem levantar dinheiro deste cofre bancÃąrio.

3. Utilizadores podem verificar o balanÃğo deste cofre bancÃąrio.

Exemplo de smart contract:

```
'use strict';

var DepositeContent = function (text) {
 if (text) {
 var o = JSON.parse(text);
 this.balance = new BigNumber(o.balance);
 this.expiryHeight = new BigNumber(o.expiryHeight);
  } else {
    this.balance = new BigNumber(0);
    this.expiryHeight = new BigNumber(0);
  }
};

DepositeContent.prototype = {
  toString: function () {
    return JSON.stringify(this);
  }
};

var BankVaultContract = function () {
  LocalContractStorage.defineMapProperty(this, "bankVault", {
    parse: function (text) {
      return new DepositeContent(text);
    },
    stringify: function (o) {
      return o.toString();
    }
  });
};

// guarda valor do contracto, apenas apÃşs `height` do bloco,␣
↪utilizadores podem levantar
BankVaultContract.prototype = {
  init: function () {
    //TODO:
  },

  save: function (height) {
    var from = Blockchain.transaction.from;
    var value = Blockchain.transaction.value;
    var bk_height = new BigNumber(Blockchain.block.height);

    var orig_deposit = this.bankVault.get(from);
    if (orig_deposit) {
      value = value.plus(orig_deposit.balance);
    }

    var deposit = new DepositeContent();
```

```
    deposit.balance = value;
    deposit.expiryHeight = bk_height.plus(height);

    this.bankVault.put(from, deposit);
  },

  takeout: function (value) {
    var from = Blockchain.transaction.from;
    var bk_height = new BigNumber(Blockchain.block.height);
    var amount = new BigNumber(value);

    var deposit = this.bankVault.get(from);
    if (!deposit) {
      throw new Error("No deposit before.");
    }

    if (bk_height.lt(deposit.expiryHeight)) {
      throw new Error("Can not takeout before expiryHeight.");
    }

    if (amount.gt(deposit.balance)) {
      throw new Error("Insufficient balance.");
    }

    var result = Blockchain.transfer(from, amount);
    if (!result) {
      throw new Error("transfer failed.");
    }
    Event.Trigger("BankVault", {
      Transfer: {
        from: Blockchain.transaction.to,
        to: from,
        value: amount.toString()
      }
    });

    deposit.balance = deposit.balance.sub(amount);
    this.bankVault.put(from, deposit);
  },
  balanceOf: function () {
    var from = Blockchain.transaction.from;
    return this.bankVault.get(from);
  },
  verifyAddress: function (address) {
    // 1-valid, 0-invalid
    var result = Blockchain.verifyAddress(address);
    return {
      valid: result == 0 ? false : true
    };
  }
```

```
};
module.exports = BankVaultContract;
```

Como pode ver no exemplo do smart contracto acima, `BankVaultContract` Ãl' um objecto protÃştipo com um mÃl'todo init(). Satisfaz a especificaÃğÃčo mais bÃąsica para escrever smart contracts como descrito prÃl'viamente. BankVaultContract implementa dois outros mÃl'todos:

- save(): Utilizadores podem depositar dinheiro no cofre do banco ao chamar o mÃl'todo save();
- takeout(): Utilizadores podem levantar dinheiro no cofre do banco ao chamar o mÃl'todo takeout();
- balanceOf(): Utilizadores podem verificar o balanÃğo do cofre do banco ao chamar o mÃl'todo balanceOf();

O cÃşdigo do contracto acima usa o objecto embutido `Blockchain` e o mÃl'todo embutido `BigNumber()`. Vamos esmiuÃğar a anÃąlise do cÃşdigo do contracto linha por linha:

save():

```
// Deposita uma quantidade no cofre

save: function (height) {
 var from = Blockchain.transaction.from;
 var value = Blockchain.transaction.value;
  var bk_height = new BigNumber(Blockchain.block.height);

  var orig_deposit = this.bankVault.get(from);
  if (orig_deposit) {
    value = value.plus(orig_deposit.balance);
  }
  var deposit = new DepositeContent();
  deposit.balance = value;
  deposit.expiryHeight = bk_height.plus(height);

  this.bankVault.put(from, deposit);
},
```

takeout ():

```
takeout: function (value) {
  var from = Blockchain.transaction.from;
  var bk_height = new BigNumber(Blockchain.block.height);
  var amount = new BigNumber(value);

  var deposit = this.bankVault.get(from);
  if (!deposit) {
    throw new Error("No deposit before.");
  }
```

```
  if (bk_height.lt(deposit.expiryHeight)) {
    throw new Error("Can not takeout before expiryHeight.");
  }

  if (amount.gt(deposit.balance)) {
    throw new Error("Insufficient balance.");
  }

  var result = Blockchain.transfer(from, amount);
  if (!result) {
    throw new Error("transfer failed.");
  }
  Event.Trigger("BankVault", {
    Transfer: {
      from: Blockchain.transaction.to,
      to: from,
      value: amount.toString()
    }
  });

  deposit.balance = deposit.balance.sub(amount);
  this.bankVault.put(from, deposit);
},
```

ImplementaÃğÃčo de smart contracts

Aprendeu a escrever um smart contract em Nebulas, e agora precisamos de o implementar na chain. Antes, foi introduzido como fazer uma transaÃğÃčo em Nebulas, e usÃąmos a interface sendTransaction() para iniciar a transferÃŁncia. Implementar um smart contract em Nebulas Ãl' efectuado ao enviar uma transaÃğÃčo, chamando a interface sendTransaction(), apenas com parÃćmetros diferentes.

```
// transaction - from, to, value, nonce, gasPrice, gasLimit,␣
↪contract
sendTransactionWithPassphrase(transaction, passphrase)
```

Obedecemos a uma convenÃğÃčo que se `from` e `to` tiverem o mesmo endereÃğo, `contract` nÃčo Ãl' nulo e `binary` Ãl' nulo, assumimos que estamos a implementar um smart contract.

- `from`: endereÃğo do criador
- `to`: endereÃğo do criador
- `value`: deve ser `"0"` ao implementar o contracto;
- `nonce`: deve ser um a somar ao nonce actual no estado da conta do criador, que pode ser obtido via `GetAccountState`.
- `gasPrice`: o gasPrice usado para implementar o smart contract, que pode ser obtido

via `GetGasPrice`, or using default values: `"1000000"`;

- `gasLimit`: o gasLimit para implementar o contracto. Pode obter uma estimativa do consumo do gÃąs para implementaÃğÃčo via `EstimateGas`, e nÃčo pode usar o valor padrÃčo. TambÃl'm pode definir um valor maior. O consumo real de gÃąs Ãl' decidido pela execuÃğÃčo da implementaÃğÃčo.
- `contract`: a informaÃğÃčo do contracto, os parÃćmetros passados na implementaÃğÃčo do contracto
  - `source`: cÃşdigo fonte do contracto
  - `sourceType`: tipo do cÃşdigo do contracto, `js`, e `ts` (correspondentes ao cÃşdigo de JavaScript e typeScript)
  - `args`: parÃćmetros para o mÃl'todo de inicializaÃğÃčo do contracto. Usa uma string vazia se nÃčo existem parÃćmetros, e usa um arranjo (array) JSON se existem.

Detailed Interface Documentation API.

Exemplo de implementaÃğÃčo de um smart contract usando curl:

```
> curl -i -H 'Accept: application/json' -X POST http://
↪localhost:8685/v1/admin/transactionWithPassphrase -H 'Content-
↪Type: application/json' -d '{"transaction": {"from":
↪"n1H4MYms9F55ehcvygwWE71J8tJC4CRr2so","to":
↪"n1H4MYms9F55ehcvygwWE71J8tJC4CRr2so", "value":"0","nonce":1,
↪"gasPrice":"20000000000","gasLimit":"2000000","contract":{"source
↪":"\"use strict\";var DepositeContent=function(text){if(text){var␣
↪o=JSON.parse(text);this.balance=new BigNumber(o.balance);this.
↪expiryHeight=new BigNumber(o.expiryHeight);}else{this.balance=new␣
↪BigNumber(0);this.expiryHeight=new BigNumber(0);}};
↪DepositeContent.prototype={toString:function(){return JSON.
↪stringify(this);}};var BankVaultContract=function()
↪{LocalContractStorage.defineMapProperty(this,\"bankVault\",
↪{parse:function(text){return new DepositeContent(text);},
↪stringify:function(o){return o.toString();}});};BankVaultContract.
↪prototype={init:function(){},save:function(height){var␣
↪from=Blockchain.transaction.from;var value=Blockchain.transaction.
↪value;var bk_height=new BigNumber(Blockchain.block.height);var␣
↪orig_deposit=this.bankVault.get(from);if(orig_deposit)
↪{value=value.plus(orig_deposit.balance);} var deposit=new␣
↪DepositeContent();deposit.balance=value;deposit.expiryHeight=bk_
↪height.plus(height);this.bankVault.put(from,deposit);},
↪takeout:function(value){var from=Blockchain.transaction.from;var␣
↪bk_height=new BigNumber(Blockchain.block.height);var
 amount=new BigNumber(value);var deposit=this.bankVault.get(from);
↪if(!deposit){throw new Error(\"No deposit before.\");} if(bk_
↪height.lt(deposit.expiryHeight)){throw new Error(\"Can not␣
↪takeout before expiryHeight.\");} if(amount.gt(deposit.balance))
↪{throw new Error(\"Insufficient balance.\");} var␣
↪result=Blockchain.transfer(from,amount);if(!result){throw new␣
↪Error(\"transfer failed.\");} Event.Trigger(\"BankVault\",
↪{Transfer:{from:Blockchain.transaction.to,to:from,value:amount.
↪toString()}});deposit.balance=deposit.balance.sub(amount);this.
↪bankVault.put(from,deposit);},balanceOf:function(){var␣
↪from=Blockchain.transaction.from;return this.bankVault.get(from);}
↪,verifyAddress:function(address){var result=Blockchain.
```

```
{"result":{"txhash":
↪"aaebb86d15ca30b86834efb600f82cbcaf2d7aaffbe4f2c8e70de53cbed17889
↪","contract_address":"n1rVLTRxQEXscTgThmbTnn2NqdWFEKwpYUM"}}
```

O valor retornado para implementar um smart contract Ãl' o endereÃğo hash da transaÃğÃčo `txhash` e o endereÃğo da implementaÃğÃčo do contracto Ãl' `contract_address`. A obtenÃğÃčo deste valor nÃčo garante o sucesso da implementaÃğÃčo do contracto, porque sendTransaction() Ãl' um processo assÃ■ncrono, que precisa de ser empacotado pelo mineiro. Tal como a transaÃğÃčo prÃl'via, a transferÃłncia nÃčo chega em tempo real, e depende da velocidade da empacotaÃğÃčo do mineiro. Portanto precisa de esperar cerca de 1 minuto, e depois pode verificar se o contracto foi implementado com sucesso, ao inquirir o endereÃğo do contracto, ou chamando este smart contract.

Verifique se a implementaÃğÃčo do contracto foi bem sucedida

Verifique o recibo da transaÃğÃčo da implementaÃğÃčo via `GetTransactionReceipt` para verificar se o contracto foi implementado com sucesso.

```
> curl -i -H 'Content-Type: application/json' -X POST http:/
↪/localhost:8685/v1/user/getTransactionReceipt -d '{"hash":
↪"aaebb86d15ca30b86834efb600f82cbcaf2d7aaffbe4f2c8e70de53cbed17889
↪"}'

{"result":{"hash":
↪"aaebb86d15ca30b86834efb600f82cbcaf2d7aaffbe4f2c8e70de53cbed17889
↪","chainId":100,"from":
↪"n1H4MYms9F55ehcvygwWE71J8tJC4CRr2so","to":
↪"n1H4MYms9F55ehcvygwWE71J8tJC4CRr2so","value":"0","nonce":
↪"1","timestamp":"1524711841","type":"deploy","data":
↪"eyJTb3VyY2VUeXBlIjoianMiLCJTb3VyY2UiOiJcInVzZSBzdHJpY3RcIjt2YXIgRGVwb3Npc
↪ZmFsc2U6dHJ1ZX07fX07bW9kdWxlLmV4cG9ydHM9QmFua1ZhdWx0Q29udHJhY3Q7IiwiQXJncy
↪","gas_price":"20000000000","gas_limit":"2000000",
↪"contract_address":"n1rVLTRxQEXscTgThmbTnn2NqdWFEKwpYUM",
↪"status":1,"gas_used":"22016"}}
```

Como mostrado acima, o estado da implementaÃğÃčo da transacÃğÃčo torna-se 1. Significa que o contracto foi implementado com sucesso.

Executar MÃl'todo de Smart Contract

A maneira de executar um mÃl'todo de smart contract em Nebulas tambÃl'm Ãl' directa, usando o mÃl'todo sendTransactionWithPassphrase() para invocar o mÃl'todo do smart contract directamente.

```
// transaction - from, to, value, nonce, gasPrice, gasLimit,␣
↪contract
sendTransactionWithPassphrase(transaction, passphrase)
```

- `from`: o endereÃğo do utilizar
- `to`: o endereÃğo do smart contract
- `value`: a quantidade de dinheiro a ser transferida pelo smart contract
- `nonce`: deve ser um a somar ao nonce actual no estado da conta do criador, que pode ser obtido via `GetAccountState`.
- `gasPrice`: o gasPrice usado para implementar o smart contract, que pode ser obtido via `GetGasPrice`, or using default values `"20000000000"`;
- `gasLimit`: o gasLimit para implementar o contracto. Pode obter uma estimativa do consumo do gÃąs para implementaÃğÃčo via `EstimateGas`, e nÃčo pode usar o valor padrÃčo. TambÃl'm pode definir um valor maior. O consumo real de gÃąs Ãl' decidido pela execuÃğÃčo da implementaÃğÃčo.
- `contract`: a informaÃğÃčo do contracto, os parÃćmetros passados na implementaÃğÃčo do contracto
- `function`: o mÃl'todo do contracto a ser chamado
- `args`: parÃćmetros para o mÃl'todo de inicializaÃğÃčo do contracto. Usa uma string vazia se nÃčo existem parÃćmetros, e usa um arranjo (array) JSON se existem.

Por exemplo, execute o mÃl'todo save() do smart contract:

```
> curl -i -H 'Accept: application/json' -X POST http://
 ↪localhost:8685/v1/admin/transactionWithPassphrase -H 'Content-
 ↪Type: application/json' -d '{"transaction":{"from":
 ↪"n1LkDi2gGMqPrjYcczUiweyP4RxTB6Go1qS","to":
 ↪"n1rVLTRxQEXscTgThmbTnn2NqdWFEKwpYUM", "value":"100","nonce":1,
 ↪"gasPrice":"20000000000","gasLimit":"2000000","contract":{
 ↪"function":"save","args":"[0]"}}, "passphrase": "passphrase"}'

{"result":{"txhash":
 ↪"5337f1051198b8ac57033fec98c7a55e8a001dbd293021ae92564d7528de3f84
 ↪","contract_address":""}}
```

Verifique que a execuÃğÃčo do mÃl'todo do contracto `save` foi bem sucedida
A execuÃğÃčo do mÃl'todo do contracto Ãl' realmente a submissÃčo de uma transaÃğÃčo na chain. Pode verificar o resultado atravÃl's da anÃąlise do recibo da transaÃğÃčo via `GetTransactionReceipt`.

```
> curl -i -H 'Content-Type: application/json' -X POST http:/
 ↪/localhost:8685/v1/user/getTransactionReceipt -d '{"hash":
 ↪"5337f1051198b8ac57033fec98c7a55e8a001dbd293021ae92564d7528de3f84
 ↪"}'

{"result":{"hash":
 ↪"5337f1051198b8ac57033fec98c7a55e8a001dbd293021ae92564d7528de3f84
 ↪","chainId":100,"from":
 ↪"n1LkDi2gGMqPrjYcczUiweyP4RxTB6Go1qS","to":
 ↪"n1rVLTRxQEXscTgThmbTnn2NqdWFEKwpYUM","value":"100","nonce
 ↪":"1","timestamp":"1524712532","type":"call","data":
 ↪"eyJGdW5jdGlvbiI6InNhdmUiLCJBcmdzIjoiWzBdIn0=","gas_price
 ↪":"20000000000","gas_limit":"2000000","contract_address":"
 ↪","status":1,"gas_used":"20361"}}
```

Como visto acima, o estado da transaÃğÃčo da implementaÃğÃčo torna-se 1. Significa que o mÃl'todo do contracto foi executado com sucesso.

Execute o mÃl'todo takeout() do smart contract:

```
> curl -i -H 'Accept: application/json' -X POST http://
 ↪localhost:8685/v1/admin/transactionWithPassphrase -H 'Content-
 ↪Type: application/json' -d '{"transaction":{"from":
 ↪"n1LkDi2gGMqPrjYcczUiweyP4RxTB6Go1qS","to":
 ↪"n1rVLTRxQEXscTgThmbTnn2NqdWFEKwpYUM", "value":"0","nonce":2,
 ↪"gasPrice":"20000000000","gasLimit":"2000000","contract":{
 ↪"function":"takeout","args":"[50]"}}, "passphrase": "passphrase"}'

{"result":{"txhash":
 ↪"46a307e9beb21f52992a7512f3705fe58ee6c1887122a1b52f5ce5fd5f536a91
 ↪","contract_address":""}}
```

Verifique que a execuÃğÃčo do mÃl'todo `takeout` do contracto sucedeu Na execuÃğÃčo do mÃl'todo de contracto `save` acima descrito, depositou 100 wei (10^-18 NAS) no smart contract `n1rVLTRxQEXscTgThmbTnn2NqdWFEKwpYUM`. Utilizando o mÃl'todo de contracto `takeout`, vai levantar 50 wei dos 100 wei depositadas. O balanÃğo do smart contract deve ser 50 wei agora.

```
> curl -i -H 'Content-Type: application/json' -X POST http:/
 ↪/localhost:8685/v1/user/accountstate -d '{"address":
 ↪"n1rVLTRxQEXscTgThmbTnn2NqdWFEKwpYUM"}'

{"result":{"balance":"50","nonce":"0","type":88}}
```

O resultado foi o esperado.

Inquirir Dados do Smart Contract

Num smart contract, a execuÃğÃčo de alguns mÃl'todos nÃčo muda nada na chain. Estes mÃl'todos foram engendrados para nos ajudar a inquirir dados de blockchains em modo somente de leitura (read-only). Em Nebulas, fornecemos um API `call` (chamada) para os utilizadores executarem esses mÃl'todos read-only.

```
// transaction - from, to, value,
 nonce, gasPrice, gasLimit, contract
call(from, to, value, nonce, gasPrice, gasLimit, contract)
```

Os parÃćmetros de `call` sÃčo os mesmos da execuÃğÃčo de um mÃl'todo de contracto.

Chame o mÃl'todo de smart contract `balanceOf`:

```
> curl -i -H 'Accept: application/json' -X POST http://
↪localhost:8685/v1/user/call -H 'Content-Type: application/json' -
↪d '{"from":"n1LkDi2gGMqPrjYcczUiweyP4RxTB6Go1qS","to":
↪"n1rVLTRxQEXscTgThmbTnn2NqdWFEKwpYUM","value":"0","nonce":3,
↪"gasPrice":"1000000","gasLimit":"2000000","contract":{"function":
↪"balanceOf","args":""}}'

{"result":{"result":"{\"balance\":\"50\",\"expiryHeight\":\"84\"}",
↪"execute_err":"","estimate_gas":"20209"}}
```

PrÃşximo passo: Tutorial 4

Armazenamento de Smart Contracts

Nebulas 101 - 04 Armazenamento de Smart Contracts

Tutorial YouTube

Antes discutiu-se como escrever smart contracts e como os implementar e invocar na Nebulas.

Agora vamos introduzir em detalhe o armazenamento de um smart contract. Os âĂIJcontractos inteligentesâĂİ de Nebulas fornecem capacidades de armazenamento de dados on-chain (na blockchain). Semelhantes ao sistema de armazenamento key-value tradicional (ex: redis), smart contracts podem ser armazenados em Nebulas ao pagar com gÃąs (gas).

LocalContractStorage

O ambiente de armazenamento de Smart Contract de Nebulas vem com armazenamento de objectos embutido, `LocalContractStorage`, que pode armazenar nÃžmeros, cadeias de caracteres (strings), e objectos JavaScript. Dados armazenados apenas podem ser usados em smart contracts. Outros contractos nÃčo serÃčo capazes de ler dados armazenados.

BÃąsicos

O API `LocalContractStorage` inclui `set`, `get` e `del`, que permitem-lhe guardar, ler, e apagar dados. O armazenamento pode conter nÃžmeros, strings, e objectos.

Armazenar Dados de `LocalContractStorage`

```
// armazenar dados. Os dados vÃčo ser guardados como strings JSON
LocalContractStorage.put(key, value);
// Ou
LocalContractStorage.set(key, value);
```

Ler Dados de `LocalContractStorage`

```
// obtÃl'm o valor da chave
LocalContractStorage.get(key);
```

Apagar Dados de `LocalContractStorage`

```
// apagar dados, dados nÃčo podem ser lidos apÃşs a sua eliminaÃğÃčo
LocalContractStorage.del(key);
// Ou
LocalContractStorage.delete(key);
```

Exemplos:

```
'use strict';

var SampleContract = function () {
};

SampleContract.prototype = {
    init: function () {
    },
    set: function (name, value) {
        // Armazenar uma string
        LocalContractStorage.set("name",name);
        // Armazenar um nÃžmero (valor)
        LocalContractStorage.set("value", value);
        // Armazenar um objecto
        LocalContractStorage.set("obj", {name:name, value:value});
    },
    get: function () {
        var name = LocalContractStorage.get("name");
        console.log("name:" + name)
        var value = LocalContractStorage.get("value");
        console.log("value:" + value)
        var obj = LocalContractStorage.get("obj");
        console.log("obj:" + JSON.stringify(obj))
    },
    del: function () {
        var result = LocalContractStorage.del("name");
        console.log("del result:" + result)
    }
};

module.exports = SampleContract;
```

AvanÃğado

AlÃl'm dos mÃl'todos bÃąsicos `set`, `get`, and `del`, `LocalContractStorage` tambÃl'm fornece mÃl'todos para vincular propriedades de smart contracts. Pode ler e escrever propriedades vinculadas directamente sem invocar interfaces `LocalContractStorage` para `get` e `set`.

Propriedades Vinculadas

InstÃćncia de um objecto,nome de campo, e descritor devem ser fornecidos para vincular propriedades.

Interface de VinculaÃğÃčo

```
// define a propriedade do objecto chamada `fieldname` para `obj`␣
↪com descritor.
// descritor padrÃčo Ãl' o descritor JSON.parse/JSON.stringify.
// retorna
defineProperty(obj, fieldName, descriptor);

// define propriedades do objecto para `obj` de `props`.
// descritor padrÃčo Ãl' o descritor JSON.parse/JSON.stringify.
// retorna
defineProperties(obj, descriptorMap);
```

Eis um exemplo para vincular propriedades num smart contract:

```
'use strict';

var SampleContract = function () {
    // O tamanho da propriedade `size` do SampleContract Ãl' uma␣
↪propriedade armazenada. LÃł e escreve para ` size` e serÃą␣
↪armazenada na chain.
    // O `descritor` estÃą definido como nulo aqui, o JSON.
↪stringify () e JSON.parse () padrÃčo serÃčo utilizados.
    LocalContractStorage.defineMapProperty(this, "size");

    // O valor da propriedade `value` do SampleContract Ãl' uma␣
↪propriedade armazenada. LÃł e escreve para `value` e serÃą␣
↪armazenada na chain.
    // Aqui estÃą uma implementaÃğÃčo de `descritor` personalizada,␣
↪armazenada como uma string, e a retornar o objecto BigNumber␣
↪durante a anÃąlise.
    LocalContractStorage.defineMapProperty(this, "value", {
        stringify: function (obj) {
            return obj.toString();
        },
        parse: function (str) {
```

```
            return new BigNumber(str);
        }
    });
    // MÃžltiplas propriedades do SampleContract estÃčo definidas␣
↪como propriedades de armazenamento em quantidade/lotes, e os␣
↪descritores correspondentes usam serializaÃğÃčo JSON por padrÃčo
    LocalContractStorage.defineProperties(this, {
        name: null,
        count: null
    });
};

module.exports = SampleContract;
```

Depois, pode ler e escrever essas propriedades directamente como o seguinte exemplo:

```
SampleContract.prototype = {
 // Usado quando o contracto Ãl' implementado inicialmente, e nÃčo␣
↪pode ser usado uma segunda vez
 init: function (name, count, size, value) {
 // Armazena os dados na chain ao implementar o contracto
 this.name = name;
 this.count = count;
 this.size = size;
 this.value = value;
 },
 testStorage: function (balance) {
 // O valor serÃą lido do armazenamento de dados na chain, e␣
↪automaticamente convertido num conjunto BigNumber de acordo com o␣
↪descritor
 var amount = this.value.plus(new BigNumber(2));
 if (amount.lessThan(new BigNumber(balance))) {
 return 0
 }
 }
};
```

VinculaÃğÃčo de Propriedades do Mapa

E mais, `LocalContractStorage` tambÃl'm fornece mÃl'todos mpara vincular propriedades do mapa. Eis um exemplo para vincular propriedades do mapa e usÃą-las num smart contract.

```
'use strict';

var SampleContract = function () {
    // Define a propriedade do `SampleContract` para `userMap`.␣
↪Dados do mapa podem depois ser armazenados na chain usando␣
↪`userMap`
```

```
    LocalContractStorage.defineMapProperty(this, "userMap");

    // Define a propriedade do `SampleContract` para␣
↪`userBalanceMap`, e define o armazenamento e serializaÃğÃčo das␣
↪funÃğÃţes de leitura.
    LocalContractStorage.defineMapProperty(this, "userBalanceMap", {
        stringify: function (obj) {
            return obj.toString();
        },
        parse: function (str) {
            return new BigNumber(str);
        }
    });

    // Define as propriedades do `SampleContract` para vÃąrios␣
↪lotes de mapas
    LocalContractStorage.defineMapProperties(this,{
        key1Map: null,
        key2Map: null
    });
};

SampleContract.prototype = {
    init: function () {
    },
    testStorage: function () {
        // Armazena os dados em userMap e serializa os dados na␣
↪chain
        this.userMap.set("robin","1");
        // Armazena os dados em userBalanceMap e guarda os dados na␣
↪chain usando uma funÃğÃčo de serializaÃğÃčo personalizada
        this.userBalanceMap.set("robin",new BigNumber(1));
    },
    testRead: function () {
        // LÃł e armazenada dados
        var balance = this.userBalanceMap.get("robin");
        this.key1Map.set("robin", balance.toString());
        this.key2Map.set("robin", balance.toString());
    }
};

module.exports = SampleContract;
```

Iterar mapa

No contracto, mapa nÃčo suporta iteradores. Se precisar de iterar o mapa, pode usar a seguinte maneira: defina dois mapas, arrayMap e dataMap. ArrayMap com um contador estrictamente crescente como chave, e dataMap com a chave de dados como chave.

```
"use strict";

var SampleContract = function () {
   LocalContractStorage.defineMapProperty(this, "arrayMap");
   LocalContractStorage.defineMapProperty(this, "dataMap");
   LocalContractStorage.defineProperty(this, "size");
};

SampleContract.prototype = {
    init: function () {
        this.size = 0;
    },

    set: function (key, value) {
        var index = this.size;
        this.arrayMap.set(index, key);
        this.dataMap.set(key, value);
        this.size +=1;
    },

    get: function (key) {
        return this.dataMap.get(key);
    },

    len:function(){
      return this.size;
    },

    iterate: function(limit, offset){
        limit = parseInt(limit);
        offset = parseInt(offset);
        if(offset>this.size){
           throw new Error("offset is not valid");
        }
        var number = offset+limit;
        if(number > this.size){
          number = this.size;
        }
        var result  = "";
        for(var i=offset;i<number;i++){
            var key = this.arrayMap.get(i);
            var object = this.dataMap.get(key);
            result += "index:"+i+" key:"+ key + " value:" +object+"_
↪";
        }
        return result;
    }

};
```

```
module.exports = SampleContract;
```

PrÃşximo passo: Tutorial 5

InteraÃğÃčo com Nebulas por API RPC

Nebulas 101 - 05 InteraÃğÃčo com Nebulas atravÃľs do API RPC

Tutorial YouTube

A chain de nÃşs Nebulas pode ser acedida e controlada remotamente atravÃľs de RPC. A chain de Nebulas fornece uma sÃľrie de APIs para obter informaÃğÃčo dos nÃşs, balanÃğo de contas, envio de transaÃğÃţes, e implementaÃğÃčo de chamadas de smart contracts.

O acesso remoto Ãă chain de Nebulas Ãľ implementado por gRPC, e pode tambÃľm ser acedido atravÃľs de HTTP por um proxy (grpc-gateway). Acesso HTTP Ãľ uma interface implementada por RESTful, com os mesmos parÃćmetros de uma interface gRPC.

API

Implentamos um servidor RPC e HTTP para fornecer um serviÃğo API em Go-Nebulas.

MÃşdulos

Todas as interfaces estÃčo divididas em dois mÃşdulos: API e Admin.

- API: Fornece interfaces que nÃčo tÃłm relaÃğÃčo com a chave privada do utilizador.
- Admin: Fornece interfaces que estÃčo relacionadas com a chave privada do utilizador.

ÃL' recomendado para todos os nÃşs de Nebulas que abram o mÃşdulo API para utilizadores pÃžblicos, e o mÃşdulo Admin para utilizadores autorizados. NÃş: (O API Admin na testnet & mainnet da Nebulas nÃčo suporta chamadas remotas)

ConfiguraÃğÃčo

Servidor RPC e HTTP pode ser configurado no ficheiro de configuraÃğÃčo de cada nÃş de Nebulas.

```
rpc {
 # Porta do serviÃğo API gRPC
 rpc_listen: ["127.0.0.1:8684"]
 # Porta do serviÃğo API HTTP
 http_listen: ["127.0.0.1:8685"]
 # MÃşdulo aberto que fornece serviÃğo http para fora
```

```
 http_module: ["api", "admin"]
}
```

Exemplo

HTTP

Aqui estÃčo alguns exemplos para invocar as interfaces HTTP usando `curl`.

GetNebState

Pode invocar `GetNebState` atravÃl's do mÃşdulo API para obter o estado corrente do nÃş local de Nebulas, incluÃ■ndo a identidade da chain, bloco da cauda (tail block), versÃţes do protocolo, entre outros.

```
> curl -i -H Accept:application/json -X GET http://localhost:8685/
↪v1/user/nebstate

{"result":{"chain_id":100,"tail":
↪"0aa1cceb7801b110fdd5217ba0a4356780c940133924d1c1a4eb60336934dab1
↪","lib":
↪"0000000000000000000000000000000000000000000000000000000000000000
↪","height":"479","protocol_version":"/neb/1.0.0","synchronized
↪":false,"version":"0.7.0"}}
```

UnlockAccount

Pode invocar `UnlockAccount` atravÃl's do mÃşdulo Admin para desbloquear uma conta na me memÃşria. Todas as contas desbloqueadas podem ser utilizadas para enviar transaÃğÃţes directamente sem senhas.

```
> curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/admin/account/unlock -d '{"address":
↪"n1NrMKTYESZRCwPFDLFKiKREzZKaN1nhQvz", "passphrase": "passphrase"}
↪'

{"result":{"result":true}}
```

RPC

Servidor RPC foi implementado com GRPC. A serializaÃğÃčo do GPRC Ãl' baseada em Protocol Buffers. Pode encontrar todos os ficheiros RPC protobuf no DirectÃşrio Nebulas RPC Protobuf.

Aqui estÃčo alguns exemplos da invocaÃğÃčo de interfaces RPC usando `golang`.

GetNebState

Podemos invocar `GetNebState` atravÃl's do mÃşdulo API para obter o estado corrente do nÃş local de Nebulas.

```
import(
 "github.com/nebulasio/go-nebulas/rpc"
 "github.com/nebulasio/go-nebulas/rpc/pb"
)

// ConfiguraÃğÃčo do endereÃğo de ligaÃğÃčo ao servidor GRPC
addr := fmt.Sprintf("127.0.0.1:%d",uint32(8684))
conn, err := grpc.Dial(addr, grpc.WithInsecure())
if err != nil {
 log.Warn("rpc.Dial() failed:", err)
}
defer conn.Close()

// Interface API para aceder a informaÃğÃčo do estado do nÃş
api := rpcpb.NewAPIServiceClient(conn)
resp, err := ac.GetNebState(context.Background(), & rpcpb.
↪GetNebStateRequest {})
if err != nil {
 log.Println("GetNebState", "failed", err)
} else {
 log.Println("GetNebState tail", resp)
}
```

LockAccount

Conta `n1NrMKTYESZRCwPFDLFKiKREzZKaN1nhQvz` foi desbloqueada apÃşs invocar `v1/admin/account/unlock` atravÃl's do pedido HTTP acima. Pode invocar `LockAccount` atravÃl's do mÃşdulo Admin para a voltar a bloquear.

```
import(
 "github.com/nebulasio/go-nebulas/rpc"
 "github.com/nebulasio/go-nebulas/rpc/pb"
)

// ConfiguraÃğÃčo do endereÃğo de ligaÃğÃčo ao servidor GRPC
addr := fmt.Sprintf("127.0.0.1:%d",uint32(8684))
conn, err := grpc.Dial(addr, grpc.WithInsecure())
if err != nil {
 log.Warn("rpc.Dial() failed:", err)
}
defer conn.Close()
```

```
// Interface Admin para aceder, bloquear endereÃğo de conta
admin := rpcpb.NewAdminServiceClient(conn)
from := "n1NrMKTYESZRCwPFDLFKiKREzZKaN1nhQvz"
resp, err = management.LockAccount(context.Background(), & rpcpb.
↪LockAccountRequest {Address: from})
if err != nil {
 log.Println("LockAccount", from, "failed", err)
} else {
 log.Println("LockAccount", from, "result", resp)
}
```

Lista API

Para mais interfaces, refira-se Ãă documentaÃğÃčo oficial:

- *API Module*
- *Admin Module*.

PrÃşximo

Bom trabalho! Vamos juntar-nos Ãă Testnet ou Mainnet oficial para desfrutar de Nebulas agora!

Join the Testnet Join the Mainnet

Todo List

Go-Nebulas

- åÿőåŁľæţŃèŕŢNebulasïijŇèőľ'NebulasæŻť'åŁååĄěåčőïijŇæŻť'åď'ŽèŕęæČĚ
- åÿőåŁľ'NebulasåijĂåŔŠåď'Žèŕ■èÍĂçŽĎsdkåžŞïijŇåęĆNode, Ruby, Python, Php, Javaç■Ľ'ç■Ľ'ïijŇæŐěåŔčæŰĞæąč
- åÿőåŁľ'NebulasåijĂåŔŠåŔŕäžď'äžŠæŰĞæąčïijŇæŐěåŔčæŰĞæąč

Research

- åÿőåŁľ'äijŸåŇŰNebulas RankçőŮæşŢïijŇæŻť'åď'ŽèŕęæČĚ
- åÿőåŁľ'NebulasåĹľ'çŤĺå¡ćåijŔåŇŰèŕĄæŸŐçŽĎæŰźåijŔèŕĄæŸŐåĚśèŕĘçőŮæşŢPoDçŽĎåőĽ'åĚĺæĂğï

Wiki

- åÿőåŁl'NebulasæŁŁwikiç£żèŕŚæĹŘä¡ăçŽĎæŕ■èŕ■ïijŇèől'æŻt'åd'ŽäžžäžĘèğčåŇžåİŮéŞ¿ïijŇäžĘèğčNe
- åÿőåŁl'NebulasåĹűä¡IJçğŚæŹőæŢŹçĺŃïijŇåÿőåŁl'æŻt'åd'ŽæšąæIJL'çijŰçĺŃèČŇæŹŕçŽĎäžžçŘĘèğčåź

Explorer

- åÿőåŁl'NebulasåőŇåŰĎåŇžåİŮæţŔèğĹåŹĺïijŇæŻt'åd'Žèŕęæ ČĚ

Wallet

- åÿőåŁl'NebulasåőŇåŰĎç¡Śéąţç L'ĹéŠśåŇĚïijŇæŻt'åd'Žèŕęæ ČĚ
- åÿőåŁl'NebulasåőŇåŰĎæąŇéİćçL'ĹéŠśåŇĚïijŇæŻt'åd'Žèŕęæ ČĚ

Crash Reporter in Nebulas

In this doc, we introduce the crash reporter in Nebulas, which is used to collect crash reports in Nebulas and send it back to Nebulas Team, so the whole community can help improving the quality of Nebulas.

Overview

We, the Nebulas Team and the Nebulas community, always try our best to ensure the stability of Nebulas, since people put their faith and properties on it. That means critical bugs are unacceptable, and we are aware of that. However, we can‘t blindly think Nebulas is stable enough or there won‘t be any bugs. Thus, we have plan B, the crash reporter, to collect crash report and send it back to Nebulas community. We hope the whole community can leverage the crash reports and keep improving Nebulas.

Using crash reporter is a very common practice. For example, Microsoft Windows includes a crash reporting service called Windows Error Reporting that prompts users to send crash reports to Microsoft for online analysis. The information goes to a central database run by Microsoft. Apple also involves a standard crash reporter in macOS, named Crash Reporter. The Crash Reporter can send the crash logs to Apple Inc, for their engineers to review. Opensource community also have their own crash reporter, like Bug Buddy for Gnome, Crashpad for Chrome, Talkback for Mozilla, and etc.

In Nebulas, the crash reporter just works like the other crash reporters. It‘s aware of the crash, collects necessary information about the crash, and sends it back the Nebulas server. The server is hosted by Nebulas, and accessible for the whole community.

As a opensource, decentralized platform, we are aware of that the crash reporter may violate some users‘ privacy concern. Thus, we remove all private information in the crash

report, like the user name, user id, user's home path and IP address. Furthermore, the crash reporter is optional and users may choose close it if users still have some concerns.

How to use it

To enable or disable the crash reporter, you need to look into the configuration file, `config.conf`, and change `enable_crash_reporter` to `true` to enable it, while `false` to disable it.

How it works

In this section, we would like to share some tech details. If you are not interested in the details, you can ignore this section.

The crash reporter is actually a daemon process, which is started by `neb`. When starting the crash reporter, `neb` will tell it the process id (pid) of `neb` process, and the crash file path. For the crash reporter, it will periodically check if the `neb` process and the crash file exists. At the time it finds the crash file, it will eliminate the private information and send it back to Nebulas.

Currently, the crash report is generated by the `stderr` output from `neb`. We'd like the work with the whole community to collect detailed information in the future.

Infrastructure

Network Protocol

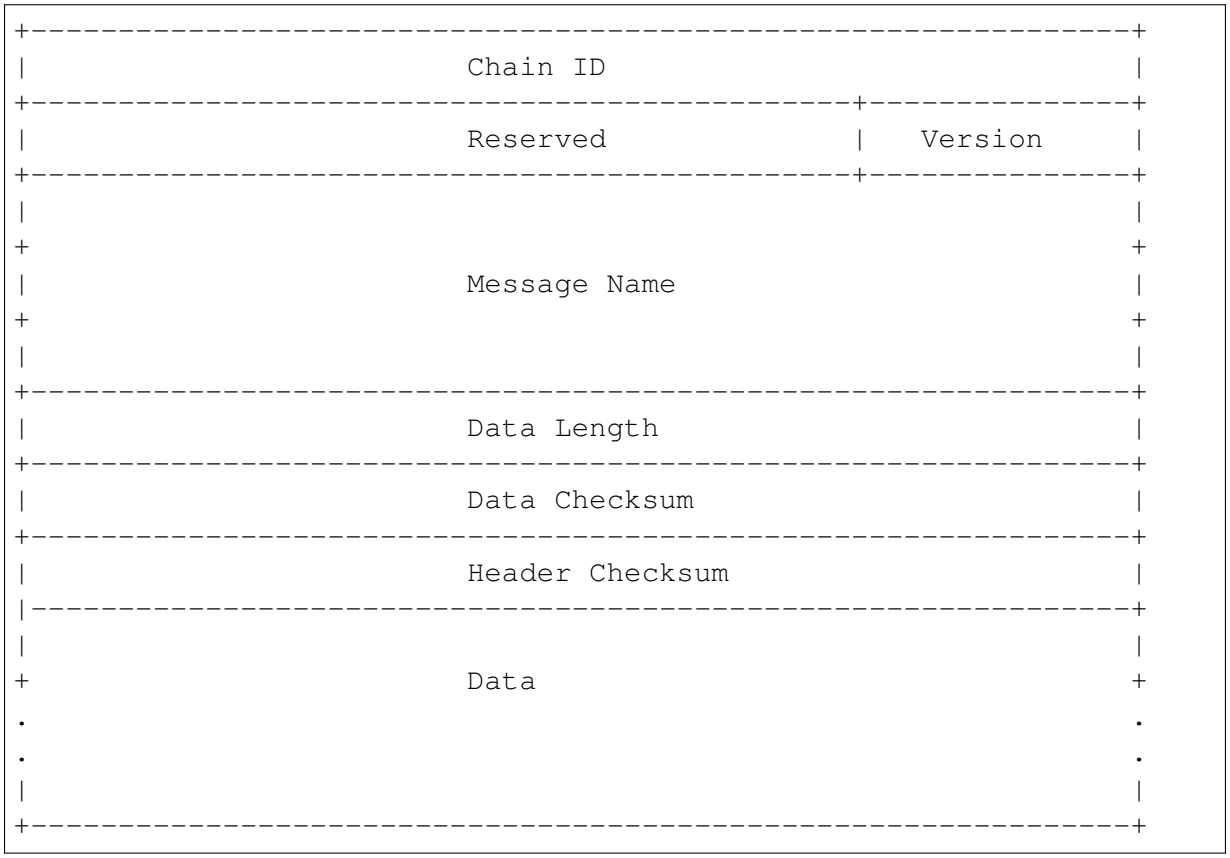
For the network protocol, there were a lot of existing solutions. However, the Nebulas Team decided to define their own wire protocol, and ensure the use of the following principles to design it:

- the protocol should be simple and straight.
- the messages can be verified before receiving all the packets, and fail early.
- the protocol should be debugging friendly, so that the developer can easily understand the raw message.

Protocol

In Nebulas, we define our own wire protocol as follows:

```
 0               1               2               3               ␣
↪(bytes)
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Magic Number                         |
```



- Magic Number: 32 bits (4 chars)
  - The protocol‘s magic number, a numerical constant or text value used to identify the protocol.
  - Default: 0x4e, 0x45, 0x42, 0x31
- Chain ID: 32 bits
  - The Chain ID is used to distinguish the test network from the main network.
- Reserved: 24 bits
  - reserved field.
  - The first bit indicates whether the network message is compressed.
  - compressed: {0x80, 0x0, 0x0}; uncompressed: {0x0, 0x0, 0x0}
- Version: 8 bits
  - The version of the Message Name.
- Message Name: 96 bits (12 chars)
  - The identification or the name of the Message.
- Data Length: 32 bits
  - The total length of the Data.
- Data Checksum: 32 bits

  - The CRC32 checksum of the Data.

- Header Checksum: 32 bits
  - The CRC32 checksum of the fields from Magic Number to Data Checksum, totally 256 bits.
- Data: variable length, max 512M.
  - The message data.

We always use Big-Endian on the message protocol.

Handshaking Messages

- Hello

the handshaking message when a peer connects to another.

```
version: 0x1

data: struct {
    string node_id  // the node id, generated by underlying libp2p.
    string client_version // the client version, x.y.z schema, eg.␣
↪0.1.0.
}
```

- OK

the response message for handshaking.

```
version: 0x1

data: struct {
    string node_id // the node id, generated by underlying libp2p.
    string node_version // the client version, x.y.z schema, eg. 0.
↪1.0.
}
```

- Bye

the message to close the connection.

```
version: 0x1
data: struct {
    string reason
}
```

Networking Messages

- NetSyncRoutes

request peers to sync route tables.

```
version: 0x1
```

- NetRoutes

contains the local route tables.

```
version: 0x1
data: struct {
    PeerID[] peer_ids // router tables.
}

struct PeerID {
    string node_id  // the node id.
}
```

Nebulas Messages

TBD.

Crypto Design Doc

Similar to Bitcoin and Ethereum, Nebulas also adopted an elliptic curve algorithm as its basic encryption algorithm for Nebulas transactions. Users' private keys will be encrypted with their passphrases and stored in a keystore.

Hash

Supports generic hash functions, like sha256, sha3256 and ripemd160 etc.

Keystore

The Nebulas Keystore is designed to manage userâĂŹs keys.

Key

The Key interface is designed to support various keys, including symmetric keys and asymmetric keys.

Provider

The Keystore provides different methods to save keys, such as *memory_provider* and *persistence_provider*. Before storage, the key has been encrypted in the keystore.

- `memory provider`: This type of provider keeps the keys in memory. After the key has been encrypted with the passphrase when user setkey or load, it is cached in memory provider.
- `persistence provider`: This type of provider serializes the encrypted key to the file. The file is compatible with EthereumâĂŹs keystore file. Users can back up the address with its privatekey in it.

Signature

The Signature interface is used to provide applications with the functionality of a digital signature algorithm. A Signature object can be used to generate and verify digital signatures.

There are two phases, in order to use a Signature object for signing data :

- Initialization: with a private key, which initializes the signature for signing (see initSign() in the source code of go-nebulas).
- Signing of all input bytes.

A Signature object can recover the public key with a signature and the plain text that was signed (see function RecoverSignerFromSignature in go-nebulas). So just comparing the from address and the address derived from the public key can verify a transaction

> Similar to the Android Keystore, TPM, TEE and hardware low level security protection will be supported as a provider later.

NVM - Nebulas Virtual Machine

NVM is one of the most important components in Nebulas. As the name implies, it provides managed virtual machine execution environments for Smart Contract and Protocol Code.

go-nebulas now support two kinds of Virtual Machines:

- V8: Chrome V8
- LLVM: Low-Level Virtual Machine

Nebulas V8 Engine

In go-nebulas, we designed and implemented the Nebulas V8 Engine based on Chrome V8.

The Nebulas V8 Engine provides a high performance sandbox for Smart Contract execution. It guarantees user deployed code is running in a managed environment, and prevents massive resource consumption on hosts. Owing to the use of Chrome V8, JavaScript and TypeScript are first-class languages for Nebulas Smart Contracts. Anyone familiar with JavaScript or TypeScript can write their own Smart Contract and run it in Nebulas V8.

The following content is an example of Smart Contract written in JavaScript:

```
"use strict";

var BankVaultContract = function() {
    LocalContractStorage.defineMapProperty(this, "bankVault");
};

// save value to contract, only after height of block, users can␣
↪takeout
BankVaultContract.prototype = {
    init:function() {},
    save:function(height) {
        var deposit = this.bankVault.get(Blockchain.transaction.
↪from);
        var value = new BigNumber(Blockchain.transaction.value);
        if (deposit != null && deposit.balance.length > 0) {
            var balance = new BigNumber(deposit.balance);
            value = value.plus(balance);
        }
        var content = {
            balance:value.toString(),
            height:Blockchain.block.height + height
        };
        this.bankVault.put(Blockchain.transaction.from, content);
    },
    takeout:function(amount) {
        var deposit = this.bankVault.get(Blockchain.transaction.
↪from);
        if (deposit == null) {
            return 0;
        }
        if (Blockchain.block.height < deposit.height) {
            return 0;
        }
        var balance = new BigNumber(deposit.balance);
        var value = new BigNumber(amount);
        if (balance.lessThan(value)) {
            return 0;
        }
        var result = Blockchain.transfer(Blockchain.transaction.
↪from, value);
        if (result > 0) {
            deposit.balance = balance.dividedBy(value).toString();
            this.bankVault.put(Blockchain.transaction.from,␣
↪deposit);
        }
        return result;
    }
};

module.exports = BankVaultContract;
```

For more information about smart contracts in Nebulas, please go to Smart Contract.

For more information about the design of the Nebulas V8 Engine, please go to Nebulas V8 Engine.

LLVM

TBD.

Nebulas V8 Engine

Nebulas V8 Engine is

LLVM Engine

TBD.

permission_control_in_smart_contract

What Is Permission Control Of Smart Contract

The permission control of a smart contract refers to whether the contract caller has permission to invoke a given function in the contract. There are two types of permission control: owner permission control, and other permission control.

Owner permissions control: Only the creator of the contract can call this method, other callers can not call the method.

Other permission control: The contract method can be invoked if the contract developer defines a conditional caller according to the contract logic. Otherwise, it cannot be invoked.

Owner Permission Control

If you want to specify an owner for a small contract and wish that some functions could only be called by the owner and no one else, you can use following lines of code in your smart contract.

```
"use strict";
var onlyOwnerContract = function () {
    LocalContractStorage.defineProperty(this, "owner");
};
onlyOwnerContract.prototype = {
  init: function() {
          this.owner=Blockchain.transaction.from;
  },
```

```
  onlyOwnerFunction: function(){
          if(this.owner==Blockchain.transaction.from){
              //your smart contract code
              return true;
          }else{
              return false;
          }
  }
};
module.exports = BankVaultContract;
```

Explanation:

The function init is only called once when the contract is deployed, so it is there that you can specify the owner of the contract.The onlyOwnerFunctiuon ensures that the function is called by the owner of contract.

Other Permission Control

In your smart contract, if you needed to specify other permission control, for example, if you needed to verify its transaction value, you could write it the following way.

```
'use strict';
var Mixin = function () {};
Mixin.UNPAYABLE = function () {
   if (Blockchain.transaction.value.lt(0)) {
       return true;
   }
   return false;
};
Mixin.PAYABLE = function () {
   if (Blockchain.transaction.value.gt(0)) {
       return true;
   }
   return false;
};
Mixin.POSITIVE = function () {
   console.log("POSITIVE");
   return true;
};
Mixin.UNPOSITIVE = function () {
   console.log("UNPOSITIVE");
   return false;
};
Mixin.decorator = function () {
   var funcs = arguments;
   if (funcs.length < 1) {
       throw new Error("mixin decorator need parameters");
   }
```

```
    return function () {
        for (var i = 0; i < funcs.length - 1; i ++) {
            var func = funcs[i];
            if (typeof func !== "function" || !func()) {
                throw new Error("mixin decorator failure");
            }
        }
        var exeFunc = funcs[funcs.length - 1];
        if (typeof exeFunc === "function") {
            exeFunc.apply(this, arguments);
        } else {
            throw new Error("mixin decorator need an executable␣
↪method");
        }
    };
};
var SampleContract = function () {
};
SampleContract.prototype = {
    init: function () {
    },
    unpayable: function () {
        console.log("contract function unpayable:", arguments);
    },
    payable: Mixin.decorator(Mixin.PAYABLE, function () {
        console.log("contract function payable:",arguments);
    }),
    contract1: Mixin.decorator(Mixin.POSITIVE, function (arg) {
        console.log("contract1 function:", arg);
    }),
    contract2: Mixin.decorator(Mixin.UNPOSITIVE, function (arg) {
        console.log("contract2 function:", arg);
    }),
    contract3: Mixin.decorator(Mixin.PAYABLE, Mixin.POSITIVE,␣
↪function (arg) {
        console.log("contract3 function:", arg);
    }),
    contract4: Mixin.decorator(Mixin.PAYABLE, Mixin.UNPOSITIVE,␣
↪function (arg) {
        console.log("contract4 function:", arg);
    })
};
module.exports = SampleContract;
```

Explanation:

Mixin.UNPAYABLE,Mixin.PAYABLE,Mixin.POSITIVE ,Mixin.UNPOSITIVE are permission control functionãĂĆThe permission control function is as follows:

- Mixin.UNPAYABLE: check the transaction sent value, if value is less than 0 return true, otherwise return false

- Mixin.PAYABLE : check the transaction sent value, if value is greater than 0 return true, otherwise return false
- Mixin.UNPOSITIVE ïijŽoutput log UNPOSITIVE
- Mixin.POSITIVE ïijŽoutput log POSITIVE

Implement permission control in Mixin.decoratorïijŽ

- check arguments: if (funcs.length < 1)
- invoke permission control function: if (typeof func !== "function" || !func())
- if permission control function success ,invoke other function: var exeFunc = funcs[funcs.length - 1]

Permission control tests in smart contracts are as follows:

- The permission control function of the contract1 is Mixin.POSITIVE. If the permission check passes, the output is printed, otherwise an error is thrown by the permission check function.

```
    contract1: Mixin.decorator(Mixin.POSITIVE, function (arg)
↪{
        console.log("contract1 function:", arg);
    })
```

- The permission control function of the contract2 is Mixin.UNPOSITIVE. If the permission check passes, the output is printed, otherwise an error is thrown by the permission check function.

```
    contract2: Mixin.decorator(Mixin.UNPOSITIVE, function␣
↪(arg) {
            console.log("contract2 function:", arg);
    })
```

- The permission control function of the contract3 is Mixin.PAYABLE, Mixin.POSITIVE. If the permission check passes, the output is printed, otherwise an error is thrown by the permission check function.

```
   contract3: Mixin.decorator(Mixin.PAYABLE, Mixin.POSITIVE,␣
↪function (arg) {
            console.log("contract3 function:", arg);
    })
```

- The permission control function of the contract4 is Mixin.PAYABLE, Mixin.UNPOSITIVE. If the permission check passes, the output is printed, otherwise an error is thrown by the permission check function.

```
    contract4: Mixin.decorator(Mixin.PAYABLE, Mixin.
↪UNPOSITIVE, function (arg) {
             console.log("contract4 function:", arg);
    })
```

Tips:

With reference to the above example, the developer needs only three steps in order to implement other permission controls:

- Implement permission control functions.
- Implement the decorator function, and the permission check is completed by the conditional statement if (typeof func !== “function“ || !func()).
- Refer to the contract1 function to implement other permission control.

2.4.6 Roteiro da Nebulas

Please visit our new Roadmap here.

Milestones

- In 2017 December, Nebulas test-net will be online.
- In 2018 Q1, Nebulas v1.0 will be released and main-net will be online (ahead of the original schedules).

v1.0 (2018 Q1)

- Fully functional blockchain, with JavaScript and TypeScript as the languages of Smart Contract.
- A user-friendly Nebulas Wallet for both desktop and mobile device to manage their own assets on Nebulas.
- A web-based Nebulas Block Explorer to let developers and users search and view all the data on Nebulas.

v2.0 (2018 Q4)

- Add Nebulas Rank (NR) to each addresses on Nebulas, help users and developers finding more values inside.
- Implement Developer Incentive Protocol (DIP) to encourage developers build more valuable decentralized applications on Nebulas.

v3.0 (2019 Q4)

- Fully functional Nebulas Force and PoD implementation.

Long term goals

- Scalability for large transaction volume.
- Subchain support.
- Zero-knowledge Proof integration.

Versions

v0.1.0 [done]

Goals

- Implement a nebulas kernel.
- In-memory blockchain with PoW consensus.
- Fully P2P network support.

Download here.

v0.2.0 [done]

Goals

- Provide (RPC) API to submit/query transaction externally.
- Implement Sync Protocol to bootstrap any nodes that join into nebulas network at any time, from any tail.

Core

- Implement transaction pool.
- Prevent record-replay attack of transaction.
- Integrate Protocol Buffer for serialization.

Net

- Refactor the design of network.
- Implement Sync Protocol.
- Implement Broadcast and Relay function.

API

- Add Balance API.
- Add Transaction API.
- Add some debugging API, eg âĂIJDump ChainâĂİ, âĂIJDump BlockâĂİ.

Crypto

- Support Ethereum-keystore file.
- Support multi key files management in KeyStore.

Download here.

v0.3.0 [done]

Goals

- Support disk storage for all blockchain data.
- Add smart contract execution engine, based on Chrome V8.

Core

- Add disk storage with a middleware of storage.
- Implement smart contract transaction.

NVM

- Integrate Chrome V8 as Smart Contract execution engine.

Download here.

v0.4.0 [done]

Goals

- Implement Gas calculating in Smart Contract Execution Engine.
- Support more API.
- Add repl in neb application.
- Add metrics and reporting capability.

Core

- Add Gas related fields in Transaction.
- Implemented Gas calculation mechanism.

NVM

- Add execution limits to V8 Engine.
- Add Gas calculation mechanism.

CMD

- Add repl in neb application

Misc

- Add more API.
- Add metrics and reporting capability.

Download here.

v0.5.0 [done]

Goals

- Prepare for test-net releasing, improve stability.

Core

- Improve stability and missing functions if we miss anything.

Consensus

- Implement DPoS consensus algorithm and keep developing PoD algorithm.

NVM

- Finalize the Gas Cost Matrix.
- Support Event liked pubsub functionality.

Misc

- Add more metrics to monitor the stability of neb applications.

Download here.

v0.6.0 [done]

Goals

- Stability improvement, performance optimization.
- Reconstruct P2P network.
- Redesign block sync logic.

Testnet

- Fix bugs & improv the performance.

Network

- Add *Stream* for single connection management.
- Add *StreamManager* for connections management.
- Implement priority message *chan*.
- Add route table persistence strategy.
- Improve strategy to process TCP packet splicing.

Log

- Add console log(CLog), printing log to both console & log files, to inform developers whatâĂŹs happening in Neb.

- Add verbose log(VLog), printing log to log files, to inform devs how Neb works in details.
- Log adjustment.

Sync

- Use chunk header hash to boost the sync performance.
- Adjust the synchronous retry logic and timeout configuration.
- Fix bugs in synchronization and add more metrics statistics.

Download here.

v0.6.1 [done]

Goals

- Improve test net compatibility.

Core

- Upgrade the storage structure of the block

Download here.

v0.8.0 [done]

Goals

- New Nebulas Block Explorer.
- New Nebulas Wallet.
- New web-based Playground tools to interactive with Nebulas.

v1.0.0 [done]

Goals

- Ready for main-net.
- Support JavaScript and TypeScript as Smart Contract Language.
- Stable and high performance blockchain system.
- Release new Nebulas Block Explorer.
- Release new Nebulas Wallet for both desktop and mobile device.
- A web-based playground tools for developer.

Download explorer.

Download wallet.

Download neb.js.

2.4.7 Desenvolvimento de DApps

Smart Contract

Linguagens

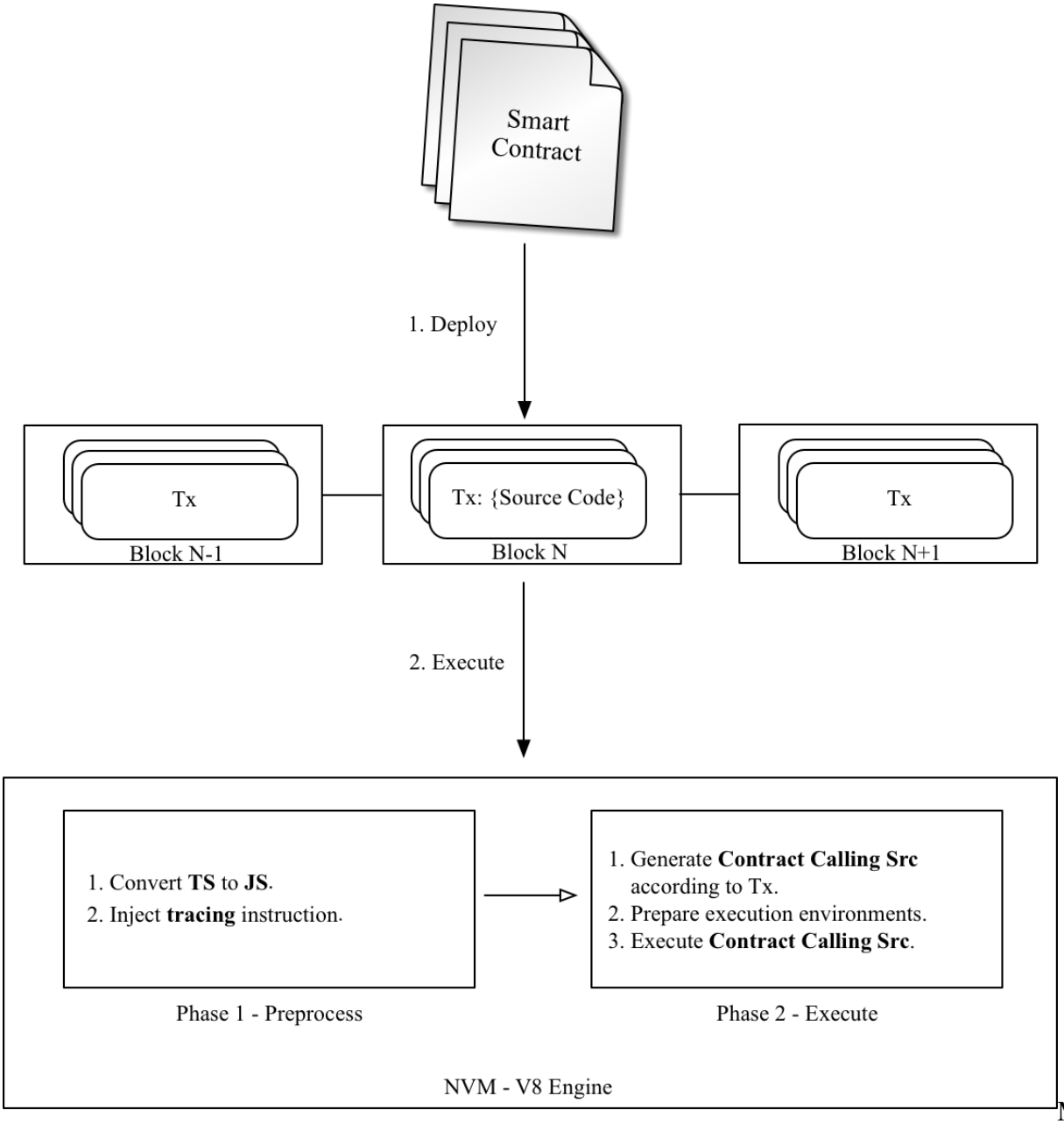
Na Nebulas suportamos duas linguagens de smart contract:

- JavaScript
- TypeScript

SÃčo suportadas pela integraÃğÃčo do Chrome V8, um motor de JavaScript desenvolvido pelo The Chromium Project para o Google Chrome e navegadores baseados no Chromium.

Modelo de ExecuÃğÃčo

O diagrama abaixo Ãl' o Modelo de ExecuÃğÃčo do Smart Contract:



Modelo de ExecuÃğÃčo do Smart Contract

1. O cÃşdigo fonte do Smart Contract e argumentos serÃčo guardados na transacÃğÃčo e implementados na Nebulas.
2. The execution of Smart Contract are divided into two phases:
   (a) Preprocesso: injecÃğÃčo de instruÃğÃčo tracing, etcetera.
   (b) Executa: gera src executÃąvel e executa-o.

Contractos

Contractos sÃčo semelhantes a classes, em linguagens orientadas a objectos. ContÃłm dados persistentes em variÃąveis de estado e funÃğÃţes que podem modificar essas variÃąveis.

Escrever um Contracto

Um contracto tem de ser um Objecto ProtÃştipo ou Classe em JavaScript ou TypeScript.

Um contracto tem de incluir uma funÃğÃčo `init`, que apenas Ãl' executada uma vez apÃşs a execuÃğÃčo. Functions, named starting with _ are `private`, can't be executed in Transaction. The others are all `public` and can be executed in Transaction.

Visto que o Contracto Ãl' executado no Chrome V8, todas as instÃćncias das variÃąveis estÃčo na memÃşria. NÃčo Ãl' aconselhÃąvel guardÃą-las todas no state trie da Nebulas. Na Nebulas, fornecemos os objectos `LocalContractStorage` e `GlobalContractStorage` de modo a permitir que desenvolvedores definam os campos que necessitam de ser guardados no state trie. Estes campos devem ser definidos no `constructor` do Contracto, antes de qualquer outra funÃğÃčo.

O seguinte Ãl' um exemplo de um contracto:

```
class Rectangle {
    constructor() {
        // define campos guardados no state trie.
        LocalContractStorage.defineProperties(this, {
            height: null,
            width: null,
        });
    }

    // init function.
    init(height, width) {
        this.height = height;
        this.width = width;
    }

    // calcula Ãąrea
    calcArea() {
        return this.height * this.width;
    }

    // verify function.
    verify(expected) {
        let area = this.calcArea();
        if (expected != area) {
            throw new Error("Error: expected " + expected + ",␣
↪actual is " + area + ".");
        }
    }
}
```

Visibilidade

Em JavaScript, nÃčo hÃą visibilidade de funÃğÃţes. Todas as funÃğÃţes definidas no objecto protÃştipo sÃčo pÃžblicas.

Na Nebulas, definimos dois tipos de visibilidade, `public` e `private`:

- `public` Todas as funÃğÃţes cujo regexp seja `^[a-zA-Z$][A-Za-z0-9_$]*$` sÃčo pÃžblicas, excepto o `init`. FunÃğÃţes pÃžblicas podem ser chamadas atravÃl's de Transaction.
- `private` Todas as funÃğÃţes cujo nome comece por _ sÃčo privadas. Uma funÃğÃčo privada apenas pode ser invocada por um funÃğÃčo pÃžblica.

Objectos Globais

linha de comandos

O mÃşdulo `console` fornece uma consola de depuraÃğÃčo que Ãl' semelhante Ãă do JavaScript fornecida por navegadores web.

A consola global pode ser usada sem usar `require('console')`.

console.info([...args])

- `...args <any>`

  A funÃğÃčo console.info() Ãl' um alias para `console.log()`.

console.log([...args])

- `...args <any>`

  Print `args` para o Logger da Nebulas, com o nÃ■vel `info`.

console.debug([...args])

- `...args <any>`

  Print `args` para o Logger da Nebulas, com o nÃ■vel `debug`.

console.warn([...args])

- `...args <any>`

  Print `args` para o Logger da Nebulas, com o nÃ■vel `warn`.

console.error([...args])

- `...args <any>`

  Print `args` para o Logger da Nebulas, com o nÃ■vel `error`.

LocalContractStorage

O mÃşdulo `LocalContractStorage` fornece capacidade de armazenamento baseada no state trie. Aceita key value pairs de cadeias de caracteres apenas. Todos os dados sÃčo guardados num state trie privado, associado com o endereÃğo do contracto, e apenas este os pode aceder.

```
interface Descriptor {
    // serialize value to string;
    stringify?(value: any): string;

    // deserialize value from string;
    parse?(value: string): any;
}

interface DescriptorMap {
    [fieldName: string]: Descriptor;
}

interface ContractStorage {
    // get and return value by key from Native Storage.
    rawGet(key: string): string;
    // set key and value pair to Native Storage,
    // return 0 for success, otherwise failure.
    rawSet(key: string, value: string): number;

    // define a object property named `fieldname` to `obj` with␣
↪descriptor.
    // default descriptor is JSON.parse/JSON.stringify descriptor.
    // return this.
    defineProperty(obj: any, fieldName: string, descriptor?:␣
↪Descriptor): any;

    // define object properties to `obj` from `props`.
    // default descriptor is JSON.parse/JSON.stringify descriptor.
    // return this.
    defineProperties(obj: any, props: DescriptorMap): any;

    // define a StorageMap property named `fieldname` to `obj` with␣
↪descriptor.
    // default descriptor is JSON.parse/JSON.stringify descriptor.
    // return this.
    defineMapProperty(obj: any, fieldName: string, descriptor?:␣
↪Descriptor): any;
```

```
    // define StorageMap properties to `obj` from `props`.
    // default descriptor is JSON.parse/JSON.stringify descriptor.
    // return this.
    defineMapProperties(obj: any, props: DescriptorMap): any;

    // delete key from Native Storage.
    // return 0 for success, otherwise failure.
    del(key: string): number;

    // get value by key from Native Storage,
    // deserialize value by calling `descriptor.parse` and return.
    get(key: string): any;

    // set key and value pair to Native Storage,
    // the value will be serialized to string by calling␣
↪`descriptor.stringify`.
    // return 0 for success, otherwise failure.
    set(key: string, value: any): number;
}

interface StorageMap {
    // delete key from Native Storage, return 0 for success,␣
↪otherwise failure.
    del(key: string): number;

    // get value by key from Native Storage,
    // deserialize value by calling `descriptor.parse` and return.
    get(key: string): any;

    // set key and value pair to Native Storage,
    // the value will be serialized to string by calling␣
↪`descriptor.stringify`.
    // return 0 for success, otherwise failure.
    set(key: string, value: any): number;
}
```

BigNumber

O modulo `BigNumber` usa bignumber.js, uma biblioteca JavaScript para precisÃčo dÃl'cimal arbitrÃąria e aritmÃl'tica nÃčo dÃl'cimal. O contracto pode usar o `BigNumber` directamente para lidar com o valor da transacÃğÃčo e transferÃłncias de outros valores.

```
var value = new BigNumber(0);
value.plus(1);
...
```

Blockchain

O modulo `Blockchain` fornece um objecto para que os contractos obtenham transacÃğÃţes e blocos executados pelo contracto actual. Adicionalmente, o NAS pode ser transferido do endereÃğo do contracto e verificaÃğÃčo do endereÃğo Ãl' fornecida.

API Blockchain:

```
// current block
Blockchain.block;

// current transaction, transaction's value/gasPrice/gasLimit auto␣
↪change to BigNumber object
Blockchain.transaction;

// transfer NAS from contract to address
Blockchain.transfer(address, value);

// verify address
Blockchain.verifyAddress(address);
```

properties:

- `block`: bloco actual para execuÃğÃčo do contracto
  - `timestamp`: marca temporal do bloco
  - `seed`: semente aleatÃşria
  - `height`: altura do bloco
- `transaction`: transacÃğÃčo actual para execuÃğÃčo do contracto
  - `hash`: hash da transacÃğÃčo
  - `from`: endereÃğo do remetente
  - `to`: endereÃğo do destinatÃąrio
  - `value`: valor da transacÃğÃčo, um objecto BigNumber para uso do contracto
  - `nonce`: nonce da transacÃğÃčo
  - `timestamp`: marca temporal da transacÃğÃčo
  - `gasPrice`: gasPrice da transacÃğÃčo, um objecto BigNumber para uso do contracto
  - `gasLimit`: gasLimit da transacÃğÃčo, um objecto BigNumber para uso do contracto
- `transfer(address, value)`: transfira NAS do contracto para o endereÃğo
  - params:
    - `address`: endereÃğo nebulas que vai receber NAS
    - `value`: valor da transferÃłncia, um objecto BigNumber

  - return:
    - `0`: sucesso da transferÃłncia
    - `1`: transferÃłncia sem sucesso
- `verifyAddress(address)`: verifica o endereÃğo
  - params:
    - `address`: endereÃğo a ser verificado
  - return:
    - `1`: endereÃğo Ãľ vÃąlido
    - `0`: endereÃğo Ãľ invÃąlido

Exemplos de como utilizar:

```
'use strict';

var SampleContract = function () {
    LocalContractStorage.defineProperties(this, {
        name: null,
        count: null
    });
    LocalContractStorage.defineMapProperty(this, "allocation");
};

SampleContract.prototype = {
    init: function (name, count, allocation) {
        this.name = name;
        this.count = count;
        allocation.forEach(function (item) {
            this.allocation.put(item.name, item.count);
        }, this);
        console.log('init: Blockchain.block.coinbase = ' +␣
↪Blockchain.block.coinbase);
        console.log('init: Blockchain.block.hash = ' + Blockchain.
↪block.hash);
        console.log('init: Blockchain.block.height = ' + Blockchain.
↪block.height);
        console.log('init: Blockchain.transaction.from = ' +␣
↪Blockchain.transaction.from);
        console.log('init: Blockchain.transaction.to = ' +␣
↪Blockchain.transaction.to);
        console.log('init: Blockchain.transaction.value = ' +␣
↪Blockchain.transaction.value);
        console.log('init: Blockchain.transaction.nonce = ' +␣
↪Blockchain.transaction.nonce);
        console.log('init: Blockchain.transaction.hash = ' +␣
↪Blockchain.transaction.hash);
    },
    transfer: function (address, value) {
```

```
        var result = Blockchain.transfer(address, value);
        console.log("transfer result:", result);
        Event.Trigger("transfer", {
            Transfer: {
                from: Blockchain.transaction.to,
                to: address,
                value: value
            }
        });
    },
    verifyAddress: function (address) {
         var result = Blockchain.verifyAddress(address);
        console.log("verifyAddress result:", result);
    }
};

module.exports = SampleContract;
```

Event

O modulo `Event` grava execuÃğÃčo de eventos no contracto. Os eventos gravados sÃčo incluÃ■dos na trie de eventos na chain, que podem ser obtidos atravÃl's do mÃl'todo `FetchEvents` no bloco com a hash da execuÃğÃčo da transacÃğÃčo. Todos os tÃşpicos de evento do contracto tÃłm o prefixo `chain.contract.` antes do tÃşpico definido no contracto.

```
Event.Trigger(topic, obj);
```

- `topic`: user-defined topic
- `obj`: JSON object

Pode ver um exemplo no `SampleContract` acima.

Math.random

- `Math.random()` returna um ponto flutuante, um nÃžmero pseudo-aleatÃşrio de domÃ■nio de 0, inclusive, atÃl', mas nÃčo incluÃ■ndo 1. O uso tÃ■pico Ãl':

```
"use strict";

var BankVaultContract = function () {};

BankVaultContract.prototype = {

    init: function () {},

    game: function(subscript){
```

```
        var arr =[1,2,3,4,5,6,7,8,9,10,11,12,13];

        for(var i = 0;i < arr.length; i++){
            var rand = parseInt(Math.random()*arr.length);
            var t = arr[rand];
            arr[rand] =arr[i];
            arr[i] = t;
        }

        return arr[parseInt(subscript)];
    },
};
module.exports = BankVaultContract;
```

- `Math.random.seed(myseed)` se necessÃąrio, pode ser usado para fazer reset da semente aleatÃşria. O argumento `myseed` tem de ser uma **string**.

  ‘‘‘js

  “use strict“;

  var BankVaultContract = function () {};

  BankVaultContract.prototype = {

```
init: function () {},

game:function(subscript, myseed){

    var arr =[1,2,3,4,5,6,7,8,9,10,11,12,13];

    console.log(Math.random());

    for(var i = 0;i < arr.length; i++){

        if (i == 8) {
            // reset random seed with `myseed`
            Math.random.seed(myseed);
        }

        var rand = parseInt(Math.random()*arr.length);
        var t = arr[rand];
        arr[rand] =arr[i];
        arr[i] = t;
    }
    return arr[parseInt(subscript)];
},
```

  };

  module.exports = BankVaultContract;

```
### Date
```js
"use strict";

var BankVaultContract = function () {};

BankVaultContract.prototype = {
    init: function () {},

    test: function(){
        var d = new Date();
        return d.toString();
    }
};

module.exports = BankVaultContract;
```

Tips:

- MÃl'todos nÃčo suportadosïijŽtoDateString(), toTimeString(), getTimezoneOffset(),toLocaleXXX().
- new Date()/Date.now() returna a marca temporal do bloco actual em milisegundos.
- getXXX returna o resultado de getUTCXXX.

accept

Este mÃl'todo visa tornar possÃ■vel o envio de uma transferÃłncia binÃąria para a conta do contracto. Como to Ãl' o endereÃğo do smart contract que foi declarado na funÃğÃčo accept() e executou correctamente, a transferÃłncia irÃą ser completada com sucesso. Se a transferÃłncia nÃčo for binÃąria, serÃą tratada como uma funÃğÃčo normal.

```
"use strict";
var DepositeContent = function (text) {
    if(text){
            var o = JSON.parse(text);
            this.balance = new BigNumber(o.balance);//ä¡ŹéćÌä£ąæĄŕ
            this.address = o.address;
    }else{
            this.balance = new BigNumber(0);
            this.address = "";
        }
};

DepositeContent.prototype = {
    toString: function () {
          return JSON.stringify(this);
    }
```
```

```
};

var BankVaultContract = function () {
    LocalContractStorage.defineMapProperty(this, "bankVault", {
         parse: function (text) {
              return new DepositeContent(text);
         },
         stringify: function (o) {
              return o.toString();
         }
    });
};

BankVaultContract.prototype = {
    init: function () {},

    save: function () {
          var from = Blockchain.transaction.from;
          var value = Blockchain.transaction.value;
          value = new BigNumber(value);
          var orig_deposit = this.bankVault.get(from);
          if (orig_deposit) {
                value = value.plus(orig_deposit.balance);
          }

          var deposit = new DepositeContent();
          deposit.balance = new BigNumber(value);
          deposit.address = from;
          this.bankVault.put(from, deposit);
    },

    accept:function(){
        this.save();
        Event.Trigger("transfer", {
            Transfer: {
                from: Blockchain.transaction.from,
                to: Blockchain.transaction.to,
                value: Blockchain.transaction.value,
            }
           });
    }

};
module.exports = BankVaultContract;
```

NRC20

Resumo

O seguinte padrÃčo permite a implementaÃğÃčo de uma API padrÃčo para tokens em contratos inteligentes.ÂăEste padrÃčo fornece funcionalidade bÃąsica para a transferÃłncia de tokens, bem como permite que os tokens sejam aprovados para que possam ser gastos por terceiros na cadeia.

MotivaÃğÃčo

Uma interface padrÃčo permite que um novo token seja facilmente criado por qualquer aplicativo: desde carteiras atÃľ trocas descentralizadas.

MÃľtodos

name

Nome. Retorna o nome do token - por exemplo `"MyToken"`.

```
// returns string, the name of the token.
function name()
```

symbol

SÃ■mbolo. Retorna o sÃ■mbolo do token.ÂăPor exemplo, `"TK"`.

```
// returns string, the symbol of the token
function symbol()
```

decimals

Decimais.ÂăRetorna o nÃžmero de decimais que o token usa - por exemplo `8`, significa dividir o valor do token `100000000` para obter sua representaÃğÃčo do utilizador.

```
// returns number, the number of decimals the token uses
function decimals()
```

totalSupply

Retorna o fornecimento total de token.

```
// returns string, the total token supply, the decimal value is␣
↪decimals* total.
function totalSupply()
```

balanceOf

Retorna o saldo da conta de um endereÃğo.

```
// returns string, the account balance of another account with␣
↪address
function balanceOf(address)
```

transfer

TransferÃłncia. Transfere a quantidade de tokens `value` para o endereÃğoÂăaddresse deve disparar o eventoÂăTransfer.ÂăA funÃğÃčo deveÂădisparar `throw`se o saldo da contaÂă`from` nÃčo tiver tokens suficientes para gastar.

> *Nota:*ÂăTransferÃłncias de 0 valores DEVEM ser tratadas como transferÃłncias normais e disparar oÂăevento `Transfer`.

```
// returns `true`, if transfer success, else throw error
function transfer(address, value)
```

transferFrom

Transfere aÂăquantidade `value` de tokens do endereÃğoÂă`from` para o endereÃğoÂă`to` e DEVE disparar oÂăevento `Transfer`.

OÂămÃl'todo `transferFrom` Ãl' usado para um fluxo de trabalho de retirada, permitindo que contratos transfiram tokens em seu nome.ÂăIsso pode ser usado, por exemplo, para permitir que um contrato transfira tokens em seu nome e / ou cobrar taxas em sub-moedas.ÂăA funÃğÃčo deveÂă`throw`, a menos que aÂă`from` conta deliberadamente autorizou o remetente da mensagem atravÃl's de algum mecanismo.

> *Nota:*ÂăTransferÃłncias de 0 valores DEVEM ser tratadas como transferÃłncias normais e disparar oÂă`Transfer` evento.

```
// returns `true`, if transfer success, else throw error
function transferFrom(from, to, value)
```

approve

Aprovar. PermiteÂăque o `spender` retire da sua conta vÃąrias vezes, atÃl'Âă`currentValue` doÂămontante `value`.ÂăSe esta funÃğÃčo for chamada no-

vamente, ela sobrescreve a permissÃčo atual comÂă`value`.

> *Nota:* Para evitar vectores de ataque, o utilizador precisa de fornecer um valor de aprovaÃğÃčo anterior e o valor padrÃčo de nÃčo aprovado Ãl' 0.

```
// returns `true`, if approve success, else throw error
function approve(spender, currentValue, value)
```

allowance

SubsÃ■dio. Retorna o valor que oÂă`spender` ainda pode ser retiradoÂădo `owner`.

```
// returns string, the value allowed to withdraw from `owner`.
function allowance(owner, spender)
```

Events

transferEvent

DEVE ser accionado quando tokens sÃčo transferidos, incluindo transferÃłncias de valor zero.

Um contrato de token que cria novos tokens DEVE accionar um evento de transferÃłncia com oÂăendereÃğo `from` definidoÂăpara o fornecimento total `totalSupply` quando os tokens sÃčo criados.

```
function transferEvent: function(status, from, to, value)
```

approveEvent

DEVE acionar em qualquer chamada para `approve(spender, currentValue, value)`.

```
function approveEvent: function(status, from, spender, value)
```

ImplementaÃğÃčo

Exemplos de implementaÃğÃţes estÃčo disponÃ■veis em

- NRC20.js

```
'use strict';

var Allowed = function (obj) {
    this.allowed = {};
```

```
    this.parse(obj);
}

Allowed.prototype = {
    toString: function () {
        return JSON.stringify(this.allowed);
    },

    parse: function (obj) {
        if (typeof obj != "undefined") {
            var data = JSON.parse(obj);
            for (var key in data) {
                this.allowed[key] = new BigNumber(data[key]);
            }
        }
    },

    get: function (key) {
        return this.allowed[key];
    },

    set: function (key, value) {
        this.allowed[key] = new BigNumber(value);
    }
}

var StandardToken = function () {
    LocalContractStorage.defineProperties(this, {
        _name: null,
        _symbol: null,
        _decimals: null,
        _totalSupply: {
            parse: function (value) {
                return new BigNumber(value);
            },
            stringify: function (o) {
                return o.toString(10);
            }
        }
    });

    LocalContractStorage.defineMapProperties(this, {
        "balances": {
            parse: function (value) {
                return new BigNumber(value);
            },
            stringify: function (o) {
                return o.toString(10);
            }
        },
```

```
        "allowed": {
            parse: function (value) {
                return new Allowed(value);
            },
            stringify: function (o) {
                return o.toString();
            }
        }
    });
};

StandardToken.prototype = {
    init: function (name, symbol, decimals, totalSupply) {
        this._name = name;
        this._symbol = symbol;
        this._decimals = decimals || 0;
        this._totalSupply = new BigNumber(totalSupply).mul(new␣
↪BigNumber(10).pow(decimals));

        var from = Blockchain.transaction.from;
        this.balances.set(from, this._totalSupply);
        this.transferEvent(true, from, from, this._totalSupply);
    },

    // Returns the name of the token
    name: function () {
        return this._name;
    },

    // Returns the symbol of the token
    symbol: function () {
        return this._symbol;
    },

    // Returns the number of decimals the token uses
    decimals: function () {
        return this._decimals;
    },

    totalSupply: function () {
        return this._totalSupply.toString(10);
    },

    balanceOf: function (owner) {
        var balance = this.balances.get(owner);

        if (balance instanceof BigNumber) {
            return balance.toString(10);
        } else {
            return "0";
```

```
        }
    },

    transfer: function (to, value) {
        value = new BigNumber(value);
        if (value.lt(0)) {
            throw new Error("invalid value.");
        }

        var from = Blockchain.transaction.from;
        var balance = this.balances.get(from) || new BigNumber(0);

        if (balance.lt(value)) {
            throw new Error("transfer failed.");
        }

        this.balances.set(from, balance.sub(value));
        var toBalance = this.balances.get(to) || new BigNumber(0);
        this.balances.set(to, toBalance.add(value));

        this.transferEvent(true, from, to, value);
    },

    transferFrom: function (from, to, value) {
        var spender = Blockchain.transaction.from;
        var balance = this.balances.get(from) || new BigNumber(0);

        var allowed = this.allowed.get(from) || new Allowed();
        var allowedValue = allowed.get(spender) || new BigNumber(0);
        value = new BigNumber(value);

        if (value.gte(0) && balance.gte(value) && allowedValue.
↪gte(value)) {

            this.balances.set(from, balance.sub(value));

            // update allowed value
            allowed.set(spender, allowedValue.sub(value));
            this.allowed.set(from, allowed);

            var toBalance = this.balances.get(to) || new␣
↪BigNumber(0);
            this.balances.set(to, toBalance.add(value));

            this.transferEvent(true, from, to, value);
        } else {
            throw new Error("transfer failed.");
        }
    },
```

```
    transferEvent: function (status, from, to, value) {
        Event.Trigger(this.name(), {
            Status: status,
            Transfer: {
                from: from,
                to: to,
                value: value
            }
        });
    },

    approve: function (spender, currentValue, value) {
        var from = Blockchain.transaction.from;

        var oldValue = this.allowance(from, spender);
        if (oldValue != currentValue.toString()) {
            throw new Error("current approve value mistake.");
        }

        var balance = new BigNumber(this.balanceOf(from));
        var value = new BigNumber(value);

        if (value.lt(0) || balance.lt(value)) {
            throw new Error("invalid value.");
        }

        var owned = this.allowed.get(from) || new Allowed();
        owned.set(spender, value);

        this.allowed.set(from, owned);

        this.approveEvent(true, from, spender, value);
    },

    approveEvent: function (status, from, spender, value) {
        Event.Trigger(this.name(), {
            Status: status,
            Approve: {
                owner: from,
                spender: spender,
                value: value
            }
        });
    },

    allowance: function (owner, spender) {
        var owned = this.allowed.get(owner);

        if (owned instanceof Allowed) {
            var spender = owned.get(spender);
```

```
            if (typeof spender != "undefined") {
                return spender.toString(10);
            }
        }
        return "0";
    }
};

module.exports = StandardToken;
```

Ferramentas

Todas as ferramentas para desenvolvimento da Nebulas, da comunidade e desenvolvedores.

- **IDE multiplataforma para Smart Contracts da Nebulas**

FunÃğÃţes Completas: web

NVM Local: Mac OS, Windows, Linux

- **nebPay**

API JavaScript de pagamentos da Nebulas. Utilizadores podem utilizÃą-lo no explorador, no PC e em mÃşvel. Podem fazer pagamentos NAS atravÃl's da extensÃčo do Chrome e das carteiras de iOS e Android.

- **Ambiente de Desenvolvimento para a Nebulas**

Ferramentas para Desenvolvimento para JavaScript

- **VS Code**
- **sublime**

Framework para Desenvolvimento de DApps

- **Nasa.js** A prezada framework do desenvolvimento de cliente de DApps da Nebulas, leve e de fÃącil utilizaÃğÃčo.
- **Ferramenta de Debugging Local de DApps da Nebulas**

Ferramentas para Desenvolvimento de Contractos

- **Ambiente de Desenvolvimento Integrado para Smart Contracts**
- **IDE para Smart Contracts da Nebulas**

Ferramenta para ImplantaÃğÃčo de Contractos

- Web-wallet
- WebExtensionWallet

Nebpay

- SDK JavaScript
- SDK iOS
- SDK Android

API Nebulas

- Go
- Python
- Java
- JavaScript
- PHP
- ruby
- NET
- unity3d
- swift

Analisador EstÃątico de CÃşdigo

- Nebulas Smart Contract Code Checker
- NebulasÂăSmart Contract Lint Tool
- Nebulas javascript/typescript smart contract static check tool

Ferramentas CLI

- Ferramenta CLI para aÂăNebulas

Ferramentas para Testes

- **NebTest visa fazer a automatizaÃğÃčo e teste unitÃąrio de smart contracts da Nebulas**

Outras

NebulasDB Ãl' uma base de dados baseada na Nebulas, descentralizada, nÃčo-relacional, e disponibiliza um cliente JS-SDK

- **A consola Ãl' de fÃącil desenvolvimento para operaÃğÃţes em dados**
- **Nebulas-Utils Ãl' um pacote para desenvolvimento da Chain da Nebulas**
- **Com base no API JS da Nebulas, pÃţeÂănebulas.js e o nebpay.js num sÃş pacote**

DApps Design Guide

TBA

Pode fazer download do PDF aqui.

Recursos de aprendizagem

Todos os recursos de aprendizagem,ÂăvÃ■deos e documentos.ÂăSerÃą bem-vindo a recomendar mais recursos da comunidade. Pode editar esta pÃągina directamente no Github.ÂăAjude os outros e aprenda em conjunto!

Documentos oficiais da Nebulas

- Papel Mauve Nebulas: Protocolo de Incentivo ao Desenvolvedor: [InglÃłs], [ChinÃłs]
- Sobre o Papel Mauve da Nebulas: Protocolo de Incentivo ao Desenvolvedor
- Nebulas Rank Yellow Paper [PortuguÃłs]
- InterpretaÃğÃčo Oficial do âĂIJNebulas Rank Yellow PaperâĂİ
- [Whitepaper TÃl'cnico: [InglÃłs], [ChinÃłs]
- Artigo nÃčo tÃl'cnico [PortuguÃłs]

Mergulhe nas Nebulas

- Mergulhe na Nebulas 1 - Uma IntroduÃğÃčo
- Mergulhe na Nebulas 2 - Um InÃ■cio RÃąpido
- Mergulhe na Nebulas 3âĂŁ-âĂŁGerÃłncia de Contas

- Mergulhe na Nebulas 4 - TransacÃğÃţes

Como construir uma DAPP em Nebulas

- Como construir uma DAPP em Nebulas: [Parte 1], [Parte 2], [Parte 3]
- Detalhes sobre o Algoritmo de classificaÃğÃčo de smart contracts: [Parte 1], [Parte 2]
- Recurso Smart New Nebulas Contract
- Reivindicar Nebulas Testnet Token Passo a Passo
- PorquÃł escolher Nebulas num Hackathon?
- Como arquitetar uma DApp usando Nuxt.js e Nebulas by Honey Thakuria
- Nebulas: JavaScript atende aos contratos inteligentes - uma introduÃğÃčo Ãă Nebulas para desenvolvedores de contratos inteligentes da Ethereum by Michal Zalecki

Como usar a Nebulas Wallet

1. CriaÃğÃčo de uma carteira NAS
2. Envio de NAS da sua carteira de Nebulas
3. Assinatura de uma transacÃğÃčo offline da Nebulas Wallet
4. Ver Carteira Informativa Nebulas Wallet
5. Verifique o estado da transacÃğÃčo na Nebulas Wallet
6. Implementar uma carteira de nebulas Smart Contract
7. InvocaÃğÃčo de smart contract na Nebulas Wallet

- Como usar o NebPay na sua DApp

AMA

- Tech Reddit AMA
- Nebulas‘ First Reddit AMA Recap
- Live Reddit AMA with Nebulas Founder Hitters Xu
- Nebulas AMA Series#1 Testnet with Nebulas Co-Founder Robin Zhong
- Nebulas AMA Series#2 Testnet with Nebulas Co-Founder Robin Zhong
- Nebulas AMA Series#3 General Question with Nebulas Co-Founder Robin Zhong
- Answers from AMA with Nebulas developer Roy Shang

RPC Overview

Overview

Chamadas de Procedimento Remoto (RPCs) criam um abstraÃğÃčo Ãžtil para a construÃğÃčo de aplicaÃğÃčo distribuÃ■das e serviÃğos.

Nebulas fornece um APi gRPC e RESTful para que os utilizadores interagirem com as Nebulas.

gRPC proporciona uma implementaÃğÃčo concreta do protocolo gRPC, por cima de HTTP/2. Estas bibliotecas permitem comunicaÃğÃčo entre clientes e servidores usando uma combinaÃğÃčo das linguagens de programaÃğÃčo suportadas.

gRPC-gateway Ãl' um plugin do protoc. LÃł a definiÃğÃčo do serviÃğo gRPC, e gera um servidor reverse-proxy que traduz o API RESTful JSON para gRPC. ÃL' entÃčo usado para mapear gRPC para HTTP.

Endpoint

Endpoints padrÃčo:

API gRPC

Podemos executar o exemplo gRPC: testing client code:

```
go run main.go
```

O cliente obtem o estado da conta do endereÃğo do remetente, faz transacÃğÃčo entre remetente e destinatÃąrio, e verifica o estado da conta do endereÃğo do remetente.

Podemos ver o output do cliente da seguinte maneira:

```
GetAccountState n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5 nonce 4 value␣
↪314283103999999999992
SendTransaction n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5 ->␣
↪n1Zn6iyyQRhqthmCfqGBzWfip1Wx8wEvtrJ value 2 txhash:
↪"2c2f5404a2e2edb651dff44a2d114a198c00614b20801e58d5b00899c8f512ae"
GetAccountState n1Zn6iyyQRhqthmCfqGBzWfip1Wx8wEvtrJ nonce 0 value 10
```

HTTP

Agora tambÃl'm fornecemos HTTP para aceder ao API RPC. O ficheiro que termina com **gw.go** Ãl' o ficheiro de mapeamento. Podemos aceder ao API RPC directamente do nosso browser. Pode actualizar o **rpc_listen** e **http_listen** em **conf/default/config.conf** para mudar a porta do RPC/HTTP, respectivamente.

Exemplo:

```
curl -i -H 'Content-Type: application/json' -X GET http://
↪localhost:8685/v1/user/nebstate
```

if success, response will be returned like this

```
{
    "result":{
        "chain_id":100,
        "tail":
↪"b10c1203d5ae6d4d069d5f520eb060f2f5fb74e942f391e7cadbc2b5148dfbcb
↪",
        "lib":
↪"da30b4ed14affb62b3719fb5e6952d3733e84e53fe6e955f8e46da503300c985
↪",
        "height":"365",
        "protocol_version":"/neb/1.0.0",
        "synchronized":false,
        "version":"0.7.0"
    }
}
```

Or, there is error form grpc, repose will carry the error message

```
{
    "error":"message..."
}
```

RPC methods

- *GetNebState*
- *GetAccountState*
- *LatestIrreversibleBlock*
- *Call*
- *SendRawTransaction*
- *GetBlockByHash*
- *GetBlockByHeight*
- *GetTransactionReceipt*
- *GetTransactionByContract*
- *GetGasPrice*
- *EstimateGas*

- *GetEventsByHash*
- *Subscribe*
- *GetDynasty*

ReferÃłncia do API RPC

GetNebState

Return o estado da neb.

Parametros

none

Returns

`chain_id` Block chain idïijŇ

- `1` : mainnet
- `1001` : testnet

`tail` neb tail hash actual.

`lib` neb lib hash actual.

`height` neb tail block height actual.

`protocol_version` a versÃčo actual do protocolo neb.

`synchronized` o estado do peer sync.

`version` versÃčo neb.

Exemplo HTTP

```
// Request
curl -i -H 'Content-Type: application/json' -X GET http://
↪localhost:8685/v1/user/nebstate

// Result
{
    "result":{
        "chain_id":100,
        "tail":
↪"b10c1203d5ae6d4d069d5f520eb060f2f5fb74e942f391e7cadbc2b5148dfbcb
↪",
```

```
        "lib":
↪"da30b4ed14affb62b3719fb5e6952d3733e84e53fe6e955f8e46da503300c985
↪",
        "height":"365",
        "protocol_version":"/neb/1.0.0",
        "synchronized":false,
        "version":"0.7.0"
    }
}
```

GetAccountState

Return o estado de uma conta. Balancete e nonce de um determinado endereÃğo.

Parametros

`address` Hex string do endereÃğo da conta.

`height` estado do block account com height. Se nÃčo for especificado, use 0 como tail height.

Returns

`balance` balanÃğo actual em unidades 1/(10^18) de nas.

`nonce` nÃžmero de transacÃğÃţes actuais.

`type` o tipo de endereÃğo. 87 significa endereÃğos normais, e 88 endereÃğos de contractos.

`height` height actual da blockchain.

`pending` transacÃğÃţes pendentes na pool de transacÃğÃţes.

Exemplo HTTP

```
// Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/user/accountstate -d '{"address":
↪"n1Z6SbjLuAEXfhX1UJvXT6BB5osWYxVg3F3"}'

// Result
{
    result {
        "balance":"9489999998980000000000"
```

```
        "nonce":51
        "type":87
            "height":"100",
        "pending":"0"
    }
}
```

LatestIrreversibleBlock

Return o Ãžltimo bloco irreversÃ■vel.

Parametros

none

Returns

`hash` Hex string do block hash.

`parent_hash` Hex string do block parent hash.

`height` block height.

`nonce` block nonce.

`coinbase` Hex string do endereÃğo da coinbase.

`timestamp` block timestamp.

`chain_id` block chain id.

`state_root` Hex string do state root.

`txs_root` Hex string do txs root.

`events_root` Hex string do event root.

`consensus_root`

- `Timestamp` tempo do estado de consenso.
- `Proposer` proponente do estado de consenso actual.
- `DynastyRoot` Hex string da dynasty root.

  `miner` o minerador deste bloco.

  `is_finality` block Ãl' finality

  `transactions` block transactions slice.
- `transaction` *GetTransactionReceipt* response info.

Exemplo HTTP

```
// Request
curl -i -H 'Content-Type: application/json' -X GET http://
↪localhost:8685/v1/user/lib

// Result
{
    "result":{
        "hash":
↪"c4a51d6241db372c1b8720e62c04426bd587e1f31054b7d04a3509f48ee58e9f
↪",
        "parent_hash":
↪"8f9f29028356d2fb2cf1291dcee85785e1c20a2145318f36c136978edb6097ce
↪",
        "height":"407",
        "nonce":"0",
        "coinbase":"n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5",
        "timestamp":"1521963660",
        "chain_id":100,
        "state_root":
↪"a77bbcd911e7ee9488b623ce4ccb8a38d9a83fc29eb5ad43009f3517f1d3e19a
↪",
        "txs_root":
↪"664671e2fda200bd93b00aaec4ab12db718212acd51b4624e8d4937003a2ab22
↪",
        "events_root":
↪"2607e32c166a3513f9effbd1dc7caa7869df5989398d0124987fa0e4d183bcaf
↪",
        "consensus_root":{
            "timestamp":"1521963660",
            "proposer":"GVeOQnYf20Ppxa2cqTrPHdpr6QH4SKs4ZKs=",
            "dynasty_root":
↪"IfTgx0o271Gg4N3cVKHe7dw3NREnlYCN8aIl8VvRXDY="
        },
        "miner": "n1WwqBXVMuYC3mFCEEuFFtAXad6yxqj4as4"
        "is_finality":false,
        "transactions":[]
    }
}
```

Call

Chama uma funÃğÃčo de um smart contract. O smart contract tem de ter sido submetido. Chamadas de mÃl'todos apenas podem ser executadas no nÃş actual.

Parametros

Os parametros do mÃl'todo de `call` sÃčo os mesmo que os do SendTransaction. AtenÃğÃčo especial:

`to` Hex string do endereÃğo do destinatÃąrio. **O valor do `to` Ãl' o endereÃğo de um contracto.**

`contract` objecto de transacÃğÃčo de contracto para chamada de smart contract.

- Sub properties(**`source` e `sourceType` nÃčo sÃčo precisos**):
- `function` a funÃğÃčo de chamada do contracto.
- `args` os parametros do contracto. O conteÃŭdo dos args Ãl' uma string JSON com um array de parametros.

Returns

`result` resultado do mÃl'todo da chamada do smart contract

`execute_err` erro de execuÃğÃčo.

`estimate_gas` estimativa de gas utilizado.

Exemplo HTTP

```
// Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/user/call -d '{"from":
↪"n1Z6SbjLuAEXfhX1UJvXT6BB5osWYxVg3F3","to":
↪"n1mL2WCZyRi1oELEugfCZoNAW3dt8QpHtJw","value":"0","nonce":3,
↪"gasPrice":"20000000000","gasLimit":"2000000","contract":{
↪"function":"transferValue","args":"[500]"}}'

// Result
{
   "result": {
       "result": "0",
       "execute_err": "insufficient balance",
       "estimate_gas": "22208"
   }
}
```

SendRawTransaction

Envia a transacÃğÃčo assinada. O valor da transacÃğÃčo assinada deve ser obtido por SignTransactionWithPassphrase.

Parameters

`data` dados assinados da transacÃğÃčo.

Returns

`txhash` Hex string do hash da transacÃğÃčo.

`contract_address` return apenas para transacÃğÃčo jÃą lanÃğada.

Exemplo HTTP

```
// Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/user/rawtransaction -d '{"data":"CiCrHtxyyIJks2/
↪RErvBBA862D6iwAaGQ9OK1NisSGAuTBIYGiY1R9Fnx0z0uPkWbPokTeBIHFFKRaosGhgzPLPtjEF5c
↪i9wAiEAAAAAAAAAAADeC2s6dkAAAoAjDd/
↪5jSBToICgZiaW5hcnlAZEoQAAAAAAAAAAAAAAAAA9CQFIQAAAAAAAAAAAAAAAAABOIFgBYkGLnnv
↪"}'

// Result
{
    "result":{
        "txhash":
↪"f37acdf93004f7a3d72f1b7f6e56e70a066182d85c186777a2ad3746b01c3b52"
    }
}
```

Exemplo de LanÃğamento de Contracto

```
// Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/user/rawtransaction -d '{"data":"CiDam3G9Sy5fV6/
↪ZcjasYPwSF39ZJDIHNB0Us94vn6p6ohIaGVfLzJ83pom1DO1gD307f1JdTVdDLzbMXO4aGhlXy8yfN
↪CEbThvI0iKcjHhgBZUB"}'

// Result
{
    "result":{
        "txhash":
↪"f37acdf93004f7a3d72f1b7f6e56e70a066182d85c186777a2ad3746b01c3b52
↪",
        "contract_address":
↪"4702b597eebb7a368ac4adbb388e5084b508af582dadde47"
    }
}
```

GetBlockByHash

Obtem informaÃğÃčo do block header atravÃl's da hash to bloco.

Parametros

`hash` Hex string do block hash.

`full_fill_transaction` se 'true' return os objectos da transacÃğÃčo. Se 'false' apenas as hashes das transacÃğÃţes.

Returns

Ver *LatestIrreversibleBlock* resposta.

Exemplo HTTP

```
// Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/user/getBlockByHash -d '{"hash":
↪"c4a51d6241db372c1b8720e62c04426bd587e1f31054b7d04a3509f48ee58e9f
↪", "full_fill_transaction":true}'

// Result
{
    "result":{
        "hash":
↪"c4a51d6241db372c1b8720e62c04426bd587e1f31054b7d04a3509f48ee58e9f
↪",
        "parent_hash":
↪"8f9f29028356d2fb2cf1291dcee85785e1c20a2145318f36c136978edb6097ce
↪",
        "height":"407",
        "nonce":"0",
        "coinbase":"n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5",
        "timestamp":"1521963660",
        "chain_id":100,
        "state_root":
↪"a77bbcd911e7ee9488b623ce4ccb8a38d9a83fc29eb5ad43009f3517f1d3e19a
↪",
        "txs_root":
↪"664671e2fda200bd93b00aaec4ab12db718212acd51b4624e8d4937003a2ab22
↪",
        "events_root":
↪"2607e32c166a3513f9effbd1dc7caa7869df5989398d0124987fa0e4d183bcaf
↪",
```

```
        "consensus_root":{
            "timestamp":"1521963660",
            "proposer":"GVeOQnYf20Ppxa2cqTrPHdpr6QH4SKs4ZKs=",
            "dynasty_root":
↪"IfTgx0o271Gg4N3cVKHe7dw3NREnlYCN8aIl8VvRXDY="
        },
        "miner": "n1WwqBXVMuYC3mFCEEuFFtAXad6yxqj4as4"
        "is_finality":false,
        "transactions":[{
            "hash":
↪"1e96493de6b5ebe686e461822ec22e73fcbfb41a6358aa58c375b935802e4145
↪",
            "chainId":100,
            "from":"n1Z6SbjLuAEXfhX1UJvXT6BB5osWYxVg3F3",
            "to":"n1orSeSMj7nn8KHHN4JcQEw3r52TVExu63r",
            "value":"10000000000000000000","nonce":"34",
            "timestamp":"1522220087",
            "type":"binary",
            "data":null,
            "gas_price":"1000000",
            "gas_limit":"2000000",
            "contract_address":"",
            "status":1,
            "gas_used":"20000"
        }]
    }
}
```


GetBlockByHeight

Obtem a informaÃğÃčo do block header atravÃl's do block height.

Parametros

`height` height da hash da transacÃğÃčo.

`full_fill_transaction` se 'true' return os objectos da transacÃğÃčo, se 'false' apenas faz return das hashes das transacÃğÃţes.

Returns

Ver *LatestIrreversibleBlock* resposta.

Exemplo HTTP

```
// Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/user/getBlockByHeight -d '{"height": 256, "full_
↪fill_transaction": true}'

// Result
{
    "result":{
        "hash":
↪"c4a51d6241db372c1b8720e62c04426bd587e1f31054b7d04a3509f48ee58e9f
↪",
        "parent_hash":
↪"8f9f29028356d2fb2cf1291dcee85785e1c20a2145318f36c136978edb6097ce
↪",
        "height":"407",
        "nonce":"0",
        "coinbase":"n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5",
        "timestamp":"1521963660",
        "chain_id":100,
        "state_root":
↪"a77bbcd911e7ee9488b623ce4ccb8a38d9a83fc29eb5ad43009f3517f1d3e19a
↪",
        "txs_root":
↪"664671e2fda200bd93b00aaec4ab12db718212acd51b4624e8d4937003a2ab22
↪",
        "events_root":
↪"2607e32c166a3513f9effbd1dc7caa7869df5989398d0124987fa0e4d183bcaf
↪",
        "consensus_root":{
            "timestamp":"1521963660",
            "proposer":"GVeOQnYf20Ppxa2cqTrPHdpr6QH4SKs4ZKs=",
            "dynasty_root":
↪"IfTgx0o271Gg4N3cVKHe7dw3NREnlYCN8aIl8VvRXDY="
        },
        "miner": "n1WwqBXVMuYC3mFCEEuFFtAXad6yxqj4as4"
        "is_finality":false,
        "transactions":[{
            "hash":
↪"1e96493de6b5ebe686e461822ec22e73fcbfb41a6358aa58c375b935802e4145
↪",
            "chainId":100,
            "from":"n1Z6SbjLuAEXfhX1UJvXT6BB5osWYxVg3F3",
            "to":"n1orSeSMj7nn8KHHN4JcQEw3r52TVExu63r",
            "value":"10000000000000000000","nonce":"34",
            "timestamp":"1522220087",
            "type":"binary",
            "data":null,
            "gas_price":"1000000",
```

```
            "gas_limit":"2000000",
            "contract_address":"",
            "status":1,
            "gas_used":"20000"
        }]
    }
}
```

GetTransactionReceipt

Obtem informaÃğÃčo do transactionReceipt atravÃl's da hash da transacÃğÃčo. Se a transacÃğÃčo nÃčo for submetida, ou apenas for submetida mas nÃčo empacotada na chain, vai fazer return do erro "not found".

Parameters

`hash` Hex string da hash da transacÃğÃčo.

Returns

`hash` Hex string da tx hash.

`chainId` Transaction chain id.

`from` Hex string do endereÃğo do remetente.

`to` Hex string do endereÃğo do destinatÃąrio.

`value` Valor da transacÃğÃčo.

`nonce` Nonce da transacÃğÃčo.

`timestamp` Timestamp da transacÃğÃčo.

`type` Tipo da transacÃğÃčo.

`data` Dados da transacÃğÃčo, return os dados do payload.

`gas_price` PreÃğo do gas da transacÃğÃčo.

`gas_limit` Limite de gas da transacÃğÃčo.

`contract_address` EndereÃğo do contracto da transacÃğÃčo.

`status` Estado da transacÃğÃčo, 0 - nÃčo sucedida, 1 - sucesso, 2 - pendente.

`gas_used` Gas usado na transacÃğÃčo.

`execute_error` Erro de execuÃğÃčo da transacÃğÃčo.

`execute_result` return o valor da funÃğÃčo do smart contract.

ObservaÃğÃčo: o comprimento dos dados de `execute_result` estÃą limitado a 255 Bytes, se quiser receber um valor superior do seu smart contract, por favor use o API `call`.

Exemplo HTTP

```
// Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/user/getTransactionReceipt -d '{"hash":
↪"cda54445ffccf4ea17f043e86e54be11b002053f9edbe30ae1fbc0437c2b6a73
↪"}'

// Result
{
    "result":{
        "hash":
↪"cda54445ffccf4ea17f043e86e54be11b002053f9edbe30ae1fbc0437c2b6a73
↪",
        "chainId":100,
        "from":"n1Z6SbjLuAEXfhX1UJvXT6BB5osWYxVg3F3",
        "to":"n1PxKRaJ5jZHXwTfgM9WqkZJJVXBxRcggEE",
        "value":"10000000000000000000",
        "nonce":"53",
        "timestamp":"1521964742",
        "type":"binary",
        "data":null,
        "gas_price":"1000000",
        "gas_limit":"20000",
        "contract_address":"",
        "status":1,
        "gas_used":"20000",
        "execute_error":"",
        "execute_result":"\"\""
    }
}
```

GetTransactionByContract

Obtem informaÃğÃčo do transactionReceipt por endereÃğo de contracto. Se o contracto nÃčo existir ou nÃčo estiver empacotado na chain, vai fazer return de erro “not found“.

Parameters

`address` Hex string do endereÃğo do contracto.

Returns

O resultado Ãl' o mesmo do GetTransactionReceipt

Exemplo HTTP

```
// Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/user/getTransactionByContract -d '{"address":
↪"n1sqDHGjYtX6rMqFoq5Tow3s3LqF4ZxBvE3"}'

// Result
{
    "result":{
        "hash":
↪"c5a45a789278f5cce9e95e8f31c1962567f58844456fed7a6eb9afcb764ca6a3
↪",
        "chainId":100,
        "from":"n1Z6SbjLuAEXfhX1UJvXT6BB5osWYxVg3F3",
        "to":"n1Z6SbjLuAEXfhX1UJvXT6BB5osWYxVg3F3",
        "value":"0",
        "nonce":"1",
        "timestamp":"1521964742",
        "type":"deploy",
        "data":
↪"eyJTb3VyY2VUeXBlIjoianMiLCJTb3VyY2UiOiJcInVzZSBzdHJpY3RcIjtcblxudmFyIENvbnRyY
↪.....
↪UmFuZG9tMlwiOiByMTIsXG4gImRlZmF1bHRTZWVkUmFuZG9tM1wiOiByMTMsXG4gICAgICAgICAgIC
↪",
        "gas_price":"1000000",
        "gas_limit":"20000",
        "contract_address":"n1sqDHGjYtX6rMqFoq5Tow3s3LqF4ZxBvE3",
        "status":1,
        "gas_used":"20000",
        "execute_error":"",
        "execute_result":"\"\""
    }
}
```

Subscribe

Return os eventos subscritos da transacÃğÃčo & bloco. O pedido Ãl' um conexÃčo "keep-alive".

Note que `subscribe` nÃčo garante a recepÃğÃčo bem-sucedida de todos os eventos, isso depende da condiÃğÃčo da rede. Por favor crie um nÃş local para usar o API `subscribe`.

Parameters

`topics` repetiÃğÃčo do nome do tÃşpico do evento, string array.

A lista de nomes de tÃşpicos:

- `chain.pendingTransaction` O tÃşpico de um transacÃğÃčo pendente numa transaction_pool.
- `chain.latestIrreversibleBlock` O tÃşpico da actualizaÃğÃčo do Ãžltimo bloco irreverssÃ■vel.
- `chain.transactionResult` O tÃşpico de execuÃğÃčo e lanÃğamento da transacÃğÃčo.
- `chain.newTailBlock` O tÃşpico de criaÃğÃčo de um novo tail block.
- `chain.revertBlock` O tÃşpico de reverter um bloco.

Returns

`topic` Nome do tÃşpico do evento subscrito.

`data` dados do evento subscrito.

Exemplo HTTP

```
// Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/user/subscribe -d '{"topics":["chain.linkBlock",
↪ "chain.pendingTransaction"]}'

// Result
{
    "result":{
        "topic":"chain.pendingTransaction",
        "data":"{
                \"chainID\":100,
                 \"hash\":\
↪"b466c7a9b667db8d15f74863a4bc60bc989566b6c3766948b2cacb45a4fbda42\
↪",
                 \"from\":\"n1Z6SbjLuAEXfhX1UJvXT6BB5osWYxVg3F3\",
                 \"to\":\"n1Z6SbjLuAEXfhX1UJvXT6BB5osWYxVg3F3\",
                 \"nonce\":6,
                 \"value\":\"0\",
                 \"timestamp\":1522215320,
                 \"gasprice\": \"20000000000\",
                 \"gaslimit\":\"20000000\",
                 \"type\":\"deploy\"}"
    }
```

```
    "result":{
        "topic":"chain.pendingTransaction",
        "data": "..."
    }
    ...
}
```

GetGasPrice

Return o gasPrice actual.

Parametros

none

Returns

`gas_price` preÃğo do gas. A unidade Ãl' 10^-18 NAS.

Exemplo HTTP

```
// Request
curl -i -H 'Content-Type: application/json' -X GET http://
↪localhost:8685/v1/user/getGasPrice

// Result
{
    "result":{
        "gas_price":"20000000000"
    }
}
```

EstimateGas

Return a estimativa de gas da transacÃğÃčo.

Parametros

Os parametros do mÃl'todo `EstimateGas` sÃčo os mesmos que os parametros de Send-Transaction.

Returns

`gas` Estimativa do gas.

`err` Mensagem de erro de execuÃğÃčo da transacÃğÃčo.

Exemplo HTTP

```
// Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/user/estimateGas -d '{"from":
↪"n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5","to":
↪"n1SAeQRVn33bamxN4ehWUT7JGdxipwn8b17", "value":
↪"1000000000000000000","nonce":1,"gasPrice":"20000000000","gasLimit
↪":"2000000"}'

// Result
{
    "result": {
        "gas":"20000",
        "err":""
    }
}
```

GetEventsByHash

Return a lista de eventos da transacÃğÃčo.

Parametros

`hash` Hex string da hash da transacÃğÃčo.

Returns

`events` a lista de eventos.

- `topic` tÃşpico dos eventos;
- `data` dados do evento.

Exemplo HTTP

```
// Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/user/getEventsByHash -d '{"hash":
↪"ec239d532249f84f158ef8ec9262e1d3d439709ebf4dd5f7c1036b26c6fe8073
↪"}'

// Result
{
    "result":{
        "events":[{
            "topic":"chain.transactionResult",
            "data":"{
                \"hash\":\
↪"d7977f96294cd232781d9c17f0f3212b48312d5ef0f556551c5cf48622759785\
↪",
                \"status\":1,
                \"gas_used\":\"22208\",
                \"error\":\"\"
            }"
        }]
    }
}
```

GetDynasty

GetDynasty obtem a dpos dynasty.

Parametros

`height` block height

Returns

`miners` string repetida do endereÃğo do minerador.

Exemplo HTTP

```
// Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/user/dynasty -d '{"height": 1}'

// Result
{
```

```
    {
        "result":{
            "miners":[
                "n1FkntVUMPAsESuCAAPK711omQk19JotBjM",
                "n1JNHZJEUvfBYfjDRD14Q73FX62nJAzXkMR",
                "n1Kjom3J4KPsHKKzZ2xtt8Lc9W5pRDjeLcW",
                "n1TV3sU6jyzR4rJ1D7jCAmtVGSntJagXZHC",
                "n1WwqBXVMuYC3mFCEEuFFtAXad6yxqj4as4",
                "n1Zn6iyyQRhqthmCfqGBzWfip1Wx8wEvtrJ"
            ]
        }
    }
}
```

RPC de GestÃčo

AlÃľm da interface do NEB API RPC, a Nebulas fornece APIs de gestÃčo adicionais. A consola Neb suporta gestÃčo de ambos APIs e interfaces de gestÃčo. RPC de gestÃčo usa a mesma porta para gRPC e HTTP, que tambÃľm vincula as interfaces do NEB API RPC.

A Nebulas fornece ambos gRPC e APIs de gestÃčo RESTful para permitir interacÃğÃčo dos utilizadores com a Nebulas. O nosso ficheiro administrativo proto define todos os admin APIs. **Recomendamos o uso da consola para aceder a interfaces de administrador, ou restringir o RPC de admin ao acesso local.**

Default management RPC Endpoint:

MÃľtodos do Management RPC

- NodeInfo
- Accounts
- NewAccount
- UnLockAccount
- LockAccount
- SignTransactionWithPassphrase
- SendTransactionWithPassphrase
- SendTransaction
- SignHash
- StartPprof
- GetConfig

ReferÃłncia do Management RPC API

NodeInfo

Return a informaÃğÃčo do nÃş p2p.

Parametros

none

Returns

`id` o ID do nÃş.

`chain_id` o block chainID.

`coninbase` coinbase.

`peer_count` nÃžmero de peers ligados.

`synchronized` estado de sincronizaÃğÃčo do nÃş.

`bucket_size` tamanho do bucket da tabela de roteamento.

`protocol_version` versÃčo do protocolo da rede.

`RouteTable*[] route_table` a routeTable da rede.

```
message RouteTable {
    string id = 1;
    repeated string address = 2;
}
```

Exemplo HTTP

```
# Request
curl -i -H 'Content-Type: application/json' -X GET http://
↪localhost:8685/v1/admin/nodeinfo

# Result
{
    "result":{
        "id":"QmP7HDFcYmJL12Ez4ZNVCKjKedfE7f48f1LAkUc3Whz4jP",
        "chain_id":100,
        "coinbase":"n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5",
        "peer_count":4,
        "synchronized":false,
        "bucket_size":64,
        "protocol_version":"/neb/1.0.0",
        "route_table":[
            {
                "id":"QmP7HDFcYmJL12Ez4ZNVCKjKedfE7f48f1LAkUc3Whz4jP
↪",
                "address":[
                    "/ip4/127.0.0.1/tcp/8680",
```

```
                    "/ip4/192.168.1.206/tcp/8680"
                ]
            },
            {
                "id":"QmUxw4PZ8kMEnHD8WaSVE92dtvdnwgufM6m5DrWemdk2M7
↪",
                "address":[
                    "/ip4/192.168.1.206/tcp/10003","/ip4/127.0.0.1/
↪tcp/10003"
                ]
            }
        ]
    }
}
```

Accounts

Return lista de contas.

Parametros

none

Returns

`addresses` account list

Exemplo HTTP

```
# Request
curl -i -H 'Content-Type: application/json' -X GET http://
↪localhost:8685/v1/admin/accounts

# Result
{
    "result":{
        "addresses":[
            "n1FkntVUMPAsESuCAAPK711omQk19JotBjM",
            "n1JNHZJEUvfBYfjDRD14Q73FX62nJAzXkMR",
            "n1Kjom3J4KPsHKKzZ2xtt8Lc9W5pRDjeLcW",
            "n1NHcbEus81PJxybnyg4aJgHAaSLDx9Vtf8",
            "n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5",
            "n1TV3sU6jyzR4rJ1D7jCAmtVGSntJagXZHC",
            "n1WwqBXVMuYC3mFCEEuFFtAXad6yxqj4as4",
            "n1Z6SbjLuAEXfhX1UJvXT6BB5osWYxVg3F3",
```

```
            "n1Zn6iyyQRhqthmCfqGBzWfip1Wx8wEvtrJ"
        ]
    }
}
```

NewAccount

NewAccount cria uma nova conta com uma password.

Parameters

`passphrase` Password da nova conta.

Returns

`address` EndereÃğo da nova conta.

Exemplo HTTP

```
# Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/admin/account/new -d '{"passphrase":"passphrase
↪"}'

# Result

{
    "result":{
        "address":"n1czGUvbQQton6KUWga4wKDLLKYDEn39mEk"
    }
}
```

UnLockAccount

UnlockAccount desbloqueia uma conta com a password. Depois do tempo de desbloqueio passar, a conta serÃą bloqueada outra vez.

Parametros

`address` EndereÃğo da conta desbloqueada.

`passphrase` Password da conta desbloqueada.

`duration` DuraÃğÃčo do desbloqueamento da conta.

Returns

`result` Resultado do desbloqueio da conta, a unidade Ãl' ns.

Exemplo HTTP

```
# Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/admin/account/unlock -d '{"address":
↪"n1czGUvbQQton6KUWga4wKDLLKYDEn39mEk","passphrase":"passphrase",
↪"duration":"1000000000"}'

# Result
{
    "result":{
        "result":true
    }
}
```

LockAccount

LockAccount bloqueia a conta.

Parameters

`address` EndereÃğo da conta bloqueada.

Returns

`result` Resultado da conta bloqueada.

Exemplo HTTP

```
# Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/admin/account/lock -d '{"address":
↪"n1czGUvbQQton6KUWga4wKDLLKYDEn39mEk"}'

# Result
{
    "result":{
        "result":true
    }
}
```

SignTransactionWithPassphrase

SignTransactionWithPassphrase assina a transacÃğÃčo. O endereÃğo `from` da conta tem de ser desbloqueado antes da chamada da assinatura.

Parametros

`transaction` usa os mesmos parametros que a SendTransaction.

`passphrase` password da conta `from`.

Returns

`data` Dados da transacÃğÃčo assinados.

Exemplo de uma transacÃğÃčo assinada normal

```
# Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/admin/sign -d '{"transaction":{"from":
↪"n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5","to":
↪"n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5", "value":
↪"1000000000000000000","nonce":1,"gasPrice":"20000000000","gasLimit
↪":"2000000"}, "passphrase":"passphrase"}'

# Result
{
    "result":{
        "data":
↪"CiBOW15yoZ+XqQbMNr4bQdJCXrBTehJKukwjcfW5eASgtBIaGVduKnw+6lM3HBXhJEzzuvv3yNdYA
↪BwhwhqUkp/
↪gEJtE4kndoc7NdSgqD26IQqa0Hjbtg1JaszAvHZiW+XH7C+Ky9XTKRJKuTOc446646d/
↪Sbz/nxQE="
    }
}
```

SendTransactionWithPassphrase

SendTransactionWithPassphrase envia transacÃğÃčo com password.

Parametros

`transaction` parametros da transacÃğÃčo, que sÃčo os mesmos da SendTransaction.

`passphrase` Password do endereÃğo `from`.

Returns

`txhash` hash da transacÃğÃčo.

`contract_address` return apenas para transacÃğÃţes de contracto lanÃğadas.

Exemplo

```
# Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/admin/transactionWithPassphrase -d '{
↪"transaction":{"from":"n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5","to":
↪"n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5", "value":
↪"1000000000000000000","nonce":1,"gasPrice":"20000000000","gasLimit
↪":"2000000"},"passphrase":"passphrase"}'

# Result
{
    "result":{
        "hash":
↪"143eac221da8079f017bd6fd6b6a08ea0623114c93c638b94334d16aae109666
↪"
```

```
        "contract_address":""
    }
}
```

SendTransaction

Envia a transacÃğÃčo. Parametros `from`, `to`, `value`, `nonce`, `gasPrice` e `gasLimit` sÃčo necessÃąrios. Se a transacÃğÃčo for para enviar um contracto, tem de especificar o `contract`.

Parametros

`from` Hex string do endereÃğo da conta do remetente.

`to` Hex string do endereÃğo da conta do destinatÃąrio.

`value` Valor da quantidade a ser enviada com nesta transacÃğÃčo.

`nonce` Nonce da transacÃğÃčo.

`gas_price` gasPrice a ser usado nesta transacÃğÃčo.

`gas_limit` gasLimit desta transacÃğÃčo.

`type` tipo do payload da transacÃğÃčo. Se o tipo for especificado, o tipo de transacÃğÃčo Ãl' determinado e o parametro correspondente tem de ser passado, ou entÃčo o tipo de transacÃğÃčo serÃą determinado de acordo com o contracto e dados binÃąrios. [optional]

- type enum:
  - `binary`: transacÃğÃčo normal com binÃąrio
  - `deploy`: lanÃğa smart contract
  - `call`: chama funÃğÃčo de smart contract

`contract` objecto de contrato de transacÃğÃčo para o lanÃğamento/chamada de smart contract. [optional]

- Sub properties:
  - `source` cÃşdigo do contracto para lanÃğar do contracto.
  - `sourceType` tipo da fonte do contracto para lanÃğar contracto. De momento suporta `js` e `ts`
    - `js` Ãl' a fonte do contracto programada em javascript.
    - `ts` Ãl' a fonte do contracto programada em typescript.
  - `function` a funÃğÃčo de chamada do contracto.
  - `args` os parametros do contracto. O conteÃždo dos args Ãl' uma string JSON de parametros num array.

`binary` qualquer dado binÃąrio com um limite igual a 64bytes. [optional]

Note:

- `from = to` quando lanÃğar um contracto, o endereÃğo `to` tem de ser igual ao endereÃğo `from`.
- `nonce` o valor Ãl' **plus one**(+1) do valor actual do nonce do endereÃğo `from`. O nonce actual pode ser obtido com o GetAccountState.
- `gasPrice` and `gasLimit` need for every transaction. We recommend taking them use GetGasPrice and EstimateGas.
- parametro `contract` apenas Ãl' preciso para o lanÃğamento ou chamada de smart contracts. Quando um smart contract Ãl' lanÃğado, a `source` e `sourceType` tem de ser especificada. Os `args` sÃčo opcionais e passados quando a inicializaÃğÃčo da funÃğÃčo leva um parametro. A field da `function` Ãl' usado no mÃl'todo de chamada.

Returns

`txhash` hash da transacÃğÃčo.

`contract_address` return apenas para transacÃğÃţes de lanÃğamento de contractos.

Exemplo de uma TransacÃğÃčo Normal

```
# Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/admin/transaction -d '{"from":
↪"n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5","to":
↪"n1SAeQRVn33bamxN4ehWUT7JGdxipwn8b17", "value":
↪"1000000000000000000","nonce":1000,"gasPrice":"20000000000",
↪"gasLimit":"2000000"}'

# Result
{
    "result":{
      "txhash":
↪"fb5204e106168549465ea38c040df0eacaa7cbd461454621867eb5abba92b4a5
↪",
      "contract_address":""
    }
}
```

Exemplo de LanÃğamento de um Smart Contract

```
# Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/admin/transaction -d '{"from":
↪"n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5","to":
↪"n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5", "value":"0","nonce":2,
↪"gasPrice":"20000000000","gasLimit":"2000000","contract":{
"source":"\"use strict\";var BankVaultContract=function()
↪{LocalContractStorage.defineMapProperty(this,\"bankVault\")};
↪BankVaultContract.prototype={init:function(){},
↪save:function(height){var deposit=this.bankVault.get(Blockchain.
↪transaction.from);var value=new BigNumber(Blockchain.transaction.
↪value);if(deposit!=null&&deposit.balance.length>0){var␣
↪balance=new BigNumber(deposit.balance);value=value.plus(balance)}
```

```
# Result
{
    "result":{
        "txhash":
↪"3a69e23903a74a3a56dfc2bfbae1ed51f69debd487e2a8dea58ae9506f572f73
↪",
        "contract_address":"n21Y7arNbUfLGL59xgnA4ouinNxyvz773NW"
    }
}
```

SignHash

SignHash assina a hash de uma mensagem.

Parameters

`address` EndereÃğo da assinatura.

`hash` Hash sha3256 da mensagem.

`alg` Algoritmo da assinatura.

Returns

`data` Dados da transacÃğÃčo assinados.

Exemplo de uma TransacÃğÃčo Assinada Normal

```
# Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/admin/sign/hash -d '{"address":
↪"n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5","hash":"W+rOKNqs/
↪tlvz02ez77yIYMCOr2EubpuNh5LvmwceI0=","alg":1}'

# Result
{
    "result":{
        "data":
↪"a7HHsLRvKTNazD1QEogY+Fre8KmBIyK+lNa4zv0Z72puFVkY9uZD6nGixGx/
↪6s1x6Baq7etGwlDNxVvHsoGWbAA="
    }
}
```

StartPprof

StartPprof starts pprof

Parametros

`listen` o endereÃğo a escutar

Returns

`result` resultado da inicializaÃğÃčo do pprof

Exemplo

```
# Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/admin/pprof -d '{"listen":"0.0.0.0:1234"}'

# Result
{
    "result":{
        "result":true
    }
}
```

GetConfig

GetConfig return a configuraÃğÃčo actual que o neb estÃą a utilizar.

Parametros

none

Returns

`config` configuraÃğÃčo do neb.

Exemplo

```
# Request
curl -i -H 'Content-Type: application/json' -X GET http://
↪localhost:8685/v1/admin/getConfig

# Result
{
    "result":{
        "config":{
            "network":{
                "seed":[],
                "listen":["0.0.0.0:8680"],
                "private_key":"conf/network/ed25519key",
                "network_id":1
            },
            "chain":{
                "chain_id":100,
                "genesis":"conf/default/genesis.conf",
                "datadir":"data.db",
                "keydir":"keydir",
                "start_mine":true,
                "coinbase":"n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5",
```

```
                "miner":"n1Zn6iyyQRhqthmCfqGBzWfip1Wx8wEvtrJ",
                "passphrase":"",
                "enable_remote_sign_server":false,
                "remote_sign_server":"",
                "gas_price":"",
                "gas_limit":"",
                "signature_ciphers":["ECC_SECP256K1"]
            },
            "rpc":{
                "rpc_listen":["127.0.0.1:8684"],
                "http_listen":["127.0.0.1:8685"],
                "http_module":["api","admin"],
                "connection_limits":0,
                "http_limits":0,
                "http_cors":[]
            },
            "stats":{
                "enable_metrics":false,
                "reporting_module":[],
                "influxdb":{
                    "host":"http://localhost:8086",
                    "port":0,
                    "db":"nebulas",
                    "user":"admin",
                    "password":"admin"
                },
                "metrics_tags":[]
            },
            "misc":null,
            "app":{
                "log_level":"debug",
                "log_file":"logs",
                "log_age":0,
                "enable_crash_report":true,
                "crash_report_url":"https://crashreport.nebulas.io",
                "pprof":{
                    "http_listen":"0.0.0.0:8888",
                    "cpuprofile":"",
                    "memprofile":""
                },
                "version":"0.7.0"
            }
        }
    }
}
```

Linha de Comandos REPL

Nebulas fornece uma linha de comandos javascript interactiva que invoca todos os mÃl'todos do API e gestÃčo RPC. A linha de comandos liga-se ao nÃş local por padrÃčo, sem ter que especificar o host manualmente..

Executar a linha de comandos

Execute a linha de comandos usando o comando:

```
./neb console
```

No caso de nÃčo ter especificado o ficheiro de configuraÃğÃčo, o terminal usa o ficheiro em `conf/default/config.conf` por padrÃčo. Se o ficheiro de configuraÃğÃčo local nÃčo estiver disponÃ■vel, ou caso queiro especificÃą-lo manualmente, o terminal comeÃğa da seguinte maneira:

```
./neb -c <config file> console
```

InteracÃğÃčo com a linha de comandos

A linha de comandos pode usar a interface `admin.setHost` para especificar que a que nÃşs se liga. Quando a linha de comandos Ãl' executada, ou o host nÃčo Ãl' especificado, o terminal estÃą a interagir com o nÃş local. **Logo, o nÃş local deve ser executado antes de abrir a linha de comandos.**

```
> admin.setHost("https://testnet.nebulas.io")
```

Conselhos: A Testnet apenas executa a interface RPC do API, portanto apenas o esquema do API estÃą disponÃ■vel.

Uso da linha de comandos

Temos os esquemas API e admin para aceder aos comandos da linha de comandos. Utilizadores podem fÃącilmente executar instruÃğÃţes ao usar a tecla `TAB`.

```
> api.
api.call                    api.getBlockByHash          api.
↪getNebState              api.subscribe
api.estimateGas             api.getBlockByHeight        api.
↪getTransactionReceipt
api.gasPrice                api.getDynasty              api.
↪latestIrreversibleBlock
api.getAccountState         api.getEventsByHash         api.
↪sendRawTransaction
```

```
> admin.
admin.accounts                          admin.nodeInfo                  ␣
 ↪    admin.signHash
admin.getConfig                         admin.sendTransaction           ␣
 ↪    admin.signTransactionWithPassphrase
admin.lockAccount                       admin.
 ↪sendTransactionWithPassphrase admin.startPprof
admin.newAccount                        admin.setHost                   ␣
 ↪    admin.unlockAccount
```

Alguns mÃl'todos de gestÃčo podem pedir palavra passe. O utilizador pode escrever a palavra passe ao invocar a interface, ou se a linha de comandos a pedir. **NÃşs recomendamos a utilizaÃğÃčo da linha de comandos para escrever a palavra pase pois esta nÃčo Ãl' visÃ■vel.**

Escreva a palavra passe directamente:

```
> admin.unlockAccount("n1UWZa8yuvRgePRPgp8a2jX4J9UwGXfHp6i",
 ↪"passphrase")
{
    "result": {
        "result": true
    }
}
```

Use o terminal:

```
> admin.unlockAccount("n1UWZa8yuvRgePRPgp8a2jX4J9UwGXfHp6i")
Unlock account n1UWZa8yuvRgePRPgp8a2jX4J9UwGXfHp6i
Passphrase:
{
    "result": {
        "result": true
    }
}
```

As interfaces com pedido de palavra passe:

```
admin.newAccount
admin.unlockAccount
admin.signHash
admin.signTransactionWithPassphrase
admin.sendTransactionWithPassphrase
```

Os parÃćmetros da linha de comandos sÃčo consistentes com os da interface RPC. NEB RPC e NEB RPC_Admin.

Fechar a linha de comandos

A linha de comandos pode ser terminada ao pressionar `ctrl-C` ou executar `exit`.

2.4.8 Tutoriais

Eis tudo o que precisa para se envolver em Nebulas.

NebulasIO

- Website: https://nebulas.io
- Livro Branco NÃčo-TÃl'cnico: https://nebulas.io/docs/NebulasWhitepaperPt.pdf
- Tech Whitepaper: https://nebulas.io/docs/NebulasTechnicalWhitepaper.pdf

Go-Nebulas

- Wiki: https://github.com/nebulasio/wiki
- Ligue-se Ãă Testnet: https://github.com/nebulasio/wiki/blob/master/testnet.md
- Ligue-se Ãă Mainnet: https://github.com/nebulasio/wiki/blob/master/mainnet.md
- Navegador: https://explorer.nebulas.io
- Tutoriais:
  - English 101
    - InstalaÃğÃčo (obrigado Cristiano)
    - Envio de TransacÃğÃţes (obrigado Cristiano)
    - Criar Smart Contract em JavaScript (obrigado Cristiano)
    - IntroduÃğÃčo de Armazenamento em Smart Contracts (obrigado Cristiano)
    - InteracÃğÃčo com Nebulas atravÃl's do API RPC (obrigado Cristiano)
  - äÿ■æŰĞ - åĚěéŮÍæŢŹçÍŃ
    - çijŰèŕŚåőL'èčĚåŔŁè£Řèąŉneb
    - åIJÍæŸ§äžŚéŞ¿äÿŁåŔŚéĂĄäžd'æŸŞ
    - ä¡£çŤÍJavaScriptçijŰåĘŹæŹžèČ¡åŘĹçžę
    - æŹžèČ¡åŘĹçžęå■ŸåĆÍåŇžäżŃçż■
    - éĂŽè£ĞRPCæŐěåŔčäÿŐæŸ§äžŚéŞ¿äžd'äžŠ

Carteira

- Carteira Web: https://github.com/nebulasio/web-wallet
  - English
    - CriaÃğÃčo de Carteira NAS

DApp

- SDK Web: https://github.com/nebulasio/neb.js
- Smart Contract: https://github.com/nebulasio/wiki/blob/master/smart_contract.md
- Protocolo PadrÃčo:
    - NRC20: https://github.com/nebulasio/wiki/blob/master/NRC20.md

Ferramentas da Comunidade

- Nebulearn: https://nebulearn.com/official-docs/go-nebulas (thanks to Tehjr)
- Demo DApp: https://github.com/15010159959/super-dictionary (thanks to ChengOrangeJu, yupnano, Kurry)
- Chrome Extension: https://github.com/ChengOrangeJu/WebExtensionWallet (thanks to ChengOrangeJu, yupnano)

Contribuir

Ficariamos felizes se escreverem tutoriais ou documentos para a Nebulas. Se criou algum, por favor submeta um pedido para nos notificar, e adicionaremos o seu nome e URL a esta pÃągina o mais rÃąpido poss+ivel.

2.4.9 Community

Eventos

Desde Junho de 2017, foram organizadas reuniÃţes e hackathons da Nebulas em 17 cidades, 8 paÃ■ses Ãă volta do mundo. VisitÃąmos a Universidade da California, Berkeley, a Universidade de Nova Iorque, a Universidade Columbia, a Universidade Harvard, a Universidade de CiÃłncias Sociais da Singapura, a Universidade de Tsinghua, a Universidade de Tongji, e muitas mais.

Historial de Eventos >

SÃčo bem vindos a organizarem reuniÃţes locais e a participar na histÃşria da Nebulas!

Comunidade da Nebulas

Dynamics

- (01/31/2019) - Nebulas Had A Fruitful Trip to Korea!
- (01/29/2019) - Week 1 Winners of Nebulas NOVA Testnet Developer Incentive Program
- (01/23/2019) - Game of Chains 2019: An Interview with Dr. Chen of Nebulas
- (01/23/2019) - Winners of Nebulas NOVA Developer Incentive Program AMA
- (01/22/2019) - Nebulas NOVA Testnet Developer Incentive Program Launches Today
- (01/17/2019) - The First Winners of Nebulas Wiki Bounty Program
- (01/11/2019) - Understanding Nebulas NOVA (Part 2)
- (01/11/2019) - Nebulas New Explorer Goes Live
- (01/10/2019) - AMA on Nebulas NOVA Developer Incentive Program
- (01/09/2019) - Nebulas Testnet Developer Incentive Program Event Guide
- (01/05/2019) - Nebulas 2018; the year in review!
- (01/04/2019) - How well do you know Nebulas NOVA?
- (01/03/2019) - Understanding Nebulas NOVA (Part 1)
- (12/31/2018) - Nebulas NOVA Testnet Released, Public Beta Testing Begins!
- (12/29/2018) - The Nebulas That IâĂŹm Looking Forward to
- (12/27/2018) - NAS nano has been upgraded to NAS nano pro
- (12/22/2018) - The Inspiration Behind the Nebulas NOVA Design
- (12/21/2018) - Why Join Nebulas
- (12/20/2018) - LetâĂŹs Check Your Core Nebulas Rank!
- (12/11/2018) - NebulersâĂŹ Thoughts on the Future of Blockchain
- (12/06/2018) - Behold: The Age of Nebulas NOVA is Upon Us!

- (11/28/2018) - Nebulas Collaborates with Key Universities at Home and Abroad - Sharing the Nebulas Wisdom
- (11/22/2018) - Public Chain TechnologyâĂŁâĂŤâĂŁthe future of blockchain?
- (11/21/2018) - Exploring the Public Chain Technology AllianceâĂŁâĂŤâĂŁBlockchainâĂŹs bridge from concept to creation!
- (11/19/2018) - To Introduce 100 Million Incremental Users to the Blockchain World
- (11/14/2018) - Nebulers are all over the world!
- (11/14/2018) - Embrace An Open and Mutually Beneficial Blockchain Ecosystem
- (11/12/2018) - DApp Development and Architecture DesignâĂŁâĂŤâĂŁInterview with Honey Thakuria
- (11/05/2018) - Let #NebulasNOVA Be a Hot Trend on TwitterïijĄ
- (11/01/2018) - Join Our Mauve Paper Reading Activity!
- (10/25/2018) - Nebulas Joining Public Chain Technology Alliance (PCTA) to Empower Developers Community
- (10/12/2018) - Nebulas Partners with UDAP to Tokenize Everything
- (09/28/2018) - Nebulas Partners with WeOne to Accelerate Global Esports Growth on the Blockchain
- (09/27/2018) - NAS nano Receives Security Audit from Knownsec
- (09/20/2018) - Liberal Radicalism: Can Quadratic Voting Be the Perfect Voting System?
- (09/04/2018) - Hello Beijing and Nebulas Team
- (08/29/2018) - Nebulas Achieving Cooperation with Knownsec, Multiple Protection and Big Data Technologies Supporting Nebulas Ecosystem Security
- (08/24/2018) - Nebulas Community Meetup ReportâĂŁâĂŤâĂŁAmbassadors Visit Beijing
- (08/09/2018) - Seeing Through The Blockchain Bubble: Sitting Down For An Interview With Nebulas.io Founder Hitters Xu
- (08/07/2018) - Blockchain Pioneers Initiate âĂIJBitsclub Vision ProgramâĂİ to Create Seamless Connection of Classical Industry and Blockchain
- (08/01/2018) - Nebulas Featured on ChinaâĂŹs Official State Website
- (08/01/2018) - Why I Love NebulasâĂŁâĂŤâĂŁPart 1: JavaScript!
- (07/31/2018) - Nebulas Partners with JOYSO to Deploy Cutting-Edge Decentralized Exchange
- (07/30/2018) - NAS Nano v2.0 is Officially Released
- (07/28/2018) - GO ! Nebulers ð§ŸĘ
- (07/28/2018) - Nebulas Incentive Program Recap

- (07/27/2018) - Nebulas Melbourne Meet-up, July 23, 2018
- (07/26/2018) - An open letter to the Nebulas community.
- (07/24/2018) - Nebulas Partners with Cocos
- (07/20/2018) - Nebulas in the Top Three of MIITâĂŹs Public Blockchain Evaluation list
- (07/17/2018) - Nebulas Partners with KingSoft Cloud (KSYUN) to Explore Blockchain Games
- (07/13/2018) - What Nebulas Research Team Says About Nebulas Rank Yellow Paper
- (07/12/2018) - Nebulas and Egretia Reach Strategic Cooperation
- (07/08/2018) - Why Choose Nebulas at a Hackathon?
- (07/05/2018) - Official Interpretation of âĂIJNebulas Rank Yellow PaperâĂİ
- (07/03/2018) - Nebulas Attended The Silicon Valley Blockchain Week
- (06/30/2018) - âĂIJThe Nebulas Rank Yellow PaperâĂİ is now public, providing the blockchain world with a more complete value measurement system.
- (06/24/2018) - Nebulas and Udacity partner to create the Global Blockchain Talent Scholarship
- (06/24/2018) - Nebulas mainnet transaction volume exceeds Ethereum, reaching nearly 700,000 for the first time
- (06/23/2018) - Cell Evolution raises 5 million RMB at more than $4.5 million USD valuation
- (06/14/2018) - Nebulas selected by ChinaâĂŹs MIIT in Global Public Chain Evaluation
- (06/14/2018) - SPIKING and NEBULAS Partner to Develop Financial Signals Search and Processing Technology for All Blockchains
- (06/09/2018) - Nebulas welcomes DeepCloud AI
- (06/01/2018) - Experienced Game Developer Bids Adieu to Ethereum and Embraces Nebulas
- (05/31/2018) - Nebulas launches DApp Store in NAS Nano update
- (05/29/2018) - Nebulas CTO Robin Zhong: âĂIJSuper nodes lead to split communitiesâĂİ, and the three key criteria for evaluating âĂŸblockchain 3.0âĂŹ
- (05/29/2018) - Latest Nebulas update paves the way for blockchain games and more!
- (05/08/2018) - Nebulas and Tencent GIS Meetup Promotes Blockchain Innovation
- (05/01/2018) - Nebulas Labs and Atlas Protocol will join Silicon Valley Entrepreneurs Festival
- (04/25/2018) - Next month: Nebulas to host its first workshops and hackathons in the US, and attend Consensus 2018 in New York City
- (04/10/2018) - LLVM x Blockchain

- (03/17/2018) - NAS Center Grand Opening & Nebulas Mainnet Launch Celebration
- (03/02/2018) - Thinking In Blockchain, a Nebulas meetup @Berkeley
- (03/02/2018) - Nebulas Enters Strategic Partnership with Dolphin Browser to Integrate the Nebulas Blockchain within its 200m User Ecosystem
- (02/16/2018) - Decentralization is the Essence of Blockchain
- (01/29/2018) - Crypto bubble 2018: Things we can do before it bursts
- (01/18/2018) - Nebulas Partners with GIFTO to Organize Blockchain Virtual Gifts for 30 Million Users

AnÃžncios

(04/08/2019) - Grupo da Comunidade da Nebulas

(04/03/2019) - A DeclaraÃğÃčo de IndependÃłncia da Nebulas: A GovernanÃğa da Nebulas Aproxima-se

(03/25/2019) - Go Nebulas: o Futuro da ColaboraÃğÃčo!

(01/29/2019) - Programa de Incentivos ao Desenvolvedor da Nebulas NOVA ComeÃğa Hoje, na Testnet

(12/27/2018) - NAS nano Foi Actualizado Para Nas Nano Pro

(12/07/2018) - Programa de Recompensas da Nebulas Wiki

(12/05/2018) - AnÃžncio: OperaÃğÃčo de ManutenÃğÃčo da Testnet da Nebulas

(11/27/2018) - Melhoria da Recompensa por Bug da Nebulas

(10/17/2018) - Processo de Candidatura para o Smartdrop de ATP ComeÃğa

(10/15/2018) - Fim do Snapshot da Mainnet da Nebulas para o Smartdrop de ATP

(10/10/2018) - AnÃžncio da Troca do Token NAS Acaba via NAS Nano

(10/08/2018) - InstruÃğÃţes para o Smartdrop de ATP

(09/25/2018) - Actualizar NAS Nano para a VersÃčo 2.2.0

(09/25/2018) - AnÃžncio da Troca do Token NAS atravÃl's do NAS Nano v.2.2.0

(09/21/2018) - Nebulas Nova: âĂIJDescubra Valor no Mundo Organizado da BlockchainâĂİ

(09/13/2018) - ApresentaÃğÃčo do ComitÃl' TÃl'cnico da Nebulas

(09/01/2018) - AnÃžncio dos EndereÃğos de NAS Bloqueados

(08/30/2018) - Aumento de Recompensa ppor Bug para Teste de FunÃğÃţes Inter-contractuais

(08/22/2018) - Recompensa para Teste do Beta PÃžblico de FunÃğÃţes Inter-contractuais na Testnet da Nebulas

(08/21/2018) - AvisoÂ de ActualizaÃğÃčo da SeguranÃğa da Mainnet

(08/08/2018) - AnÃžncio do Ajuste da DistribuiÃğÃčo do NAS Reservado para a Equipa da Nebulas

(06/29/2018) - AnÃžncio da Troca de Tokens NAS para Mainnet da Nebulas

(06/28/2018) - Livro Amarelo do Nebulas RankâĂŁâĂŤâĂŁBlockchain 3.0 Primer

(06/20/2018) - ActualizaÃğÃčo da Mainnet da Nebulas

(05/11/2018) - ExplicaÃğÃčo Suplementar para a Troca de Tokens NAS

(04/30/2018) - AnÃžncio para a Mainnet NAS: Troca de Tokens ComeÃğa nos Mercados

(04/25/2018) - DeclaraÃğÃčo sobre SeguranÃğa de Activos da Equipa TÃl'cnica da Nebulas

(04/13/2018) - AnÃžncio Importante do programa de Lock Up Bonus da Nebulas

(03/29/2018) - NotÃ■cia da FundaÃğÃčo Nebulas

(03/27/2018) - AnÃžncio da ActualizaÃğÃčo de Testnet da Nebulas

(12/13/2017) - Aviso Legal Sobre Pools Privadas para a PrÃl'-Venda de NAS

(12/13/2017) - Novas Regras de PrecificaÃğÃčo para a PrÃl'-Venda de NAS

(11/24/2017) - InÃ■cio do âĂIJNAS Token Bonus ProgramâĂİ

RelatÃşrio Semanal

Pode ver todos os relatÃşrios semanais da Nebulas aqui.

- Weekly Report #75 (04/08/2019) - Nebulas Bi-Weekly Development Commits #76
- Weekly Report #75 (04/01/2019) - Nebulas Weekly Report #75 Go Nebulas has Liftoff!
- Weekly Report #74 (03/25/2019) - Nebulas Bi-Weekly Development Commits #74
- Weekly Report #73 (03/18/2019) - Nebulas Weekly Report #73: PCTA meetup and AMA sessions
- Weekly Report #72 (03/11/2019) - Nebulas Bi-Weekly Development Commits #72
- Weekly Report #71 (03/04/2019) - Nebulas Weekly Report #71: Nebulas community development roadmap officially released
- Weekly Report #70 (02/25/2019) - Nebulas Bi-Weekly Development Commits #70
- Weekly Report #69 (02/19/2019) - Nebula Weekly Report #69: Nebulas Actively Explore Japan & Hong Kong Market in The New Year
- Weekly Report #68 (02/12/2019) - Nebulas Bi-Weekly Development Commits #68
- Weekly Report #67 (02/04/2019) - Nebulas Bi-Weekly Community Dynamics#67
- Weekly Report # 66 (01/29/2019) - Development: Nebulas Bi-Weekly Development Commits #66

- Weekly Report # 65 (01/22/2019) - Community: Nebulas Bi-Weekly Community Dynamics#65
- Weekly Report #64 (01/15/2019) -Development : We are currently working on new API features and more test cases via the testnet
- Weekly Report #63 (01/07/2019) -Community : Nebulas Bi-Weekly Community Dynamics#63
- Weekly Report #62 (12/31/2018) -Development : We finished all developments of Nebulas Nova features
- Weekly Report #61 (12/24/2018) -Community : Interview with Nebulas Team Series
- Weekly Report #60 (12/17/2018) -Development : The implementation and integration testing of the on-chain NR algorithm have been completed
- Weekly Report #59 (12/10/2018) -Community ïijŽThe Roadmap of Autonomous Metanet was Officially Released
- Weekly Report #58 (12/03/2018) -Development ïijŽCommunity can now submit their NRC 20 project to NAS nano
- Weekly Report #57 (11/26/2018) -Community : Nebulas Technical Committee Meeting Minutes(2018.11.21)
- Weekly Report #56 (11/19/2018) -Development : NBRE Has New Development
- Weekly Report #55 (11/12/2018) -Community : PCTA Launch Press Conference Successfully Held
- Weekly Report #54 (11/05/2018) -Development : Adding ATP Transfer Support
- Weekly Report #53 (10/29/2018) -Community : Nebulas Joined PCTA As One of Its First Partners
- Weekly Report #52 (10/22/2018) -Development : Improving Some Functionalities of NBRE
- Weekly Report #51 (10/15/2018) -Community : NAS Token Swap via NAS nano (v2.2.0) has Ended
- Weekly Report #50 (10/08/2018) -Development : Compeleting the Basic Implementation of NBRE
- Weekly Report #49 (10/01/2018) -Community : Nebulas Nova Development Roadmap was Announced
- Weekly Report #48 (09/24/2018) -Development : Finishing Functional Verification of NBRE
- Weekly Report #47 (09/17/2018) -Community : Nebulas Team Establishes of Nebulas Technical Committee
- Weekly Report #46 (09/10/2018) -Development : Nebulas Rank has been Realized and Open Sourced

- Weekly Report #45 (09/03/2018) -Community : Nebulas Team Announced Unreleased Locking NAS Addresses
- Weekly Report #44 (08/27/2018) -Development : NAS nano is Back to Apple Store Again
- Weekly Report #43 (08/20/2018) -Community : Nebulas Inter-contract Call Function Starts Open Beta
- Weekly Report #42 (08/13/2018) -Announcing the Adjustment of Reserved NAS Distribution to the Nebulas Team
- Weekly Report #41 (08/06/2018) -Nebulas Founders Initiate Bitsclub Vision Program
- Weekly Report #40 (07/31/2018) -Nebulas Incentive Program Season 1 Recap
- Weekly Report #39 (07/24/2018) -Nebulas in the Top Three of MIITâĂŹs Public Blockchain Evaluation list
- Weekly Report #38 (07/17/2018) -The 50 Monthly Super Contributors Were Announced
- Weekly Report #37 (07/10/2018) -Official Interpretation of the Nebulas Rank Yellow Paper
- Weekly Report #36 (07/03/2018) -The âĂIJNebulas RankâĂİ Yellow Paper is now public
- Weekly Report #35 (06/26/2018) -Nebulas participates in the Silicon Valley Blockchain Week hackathon
- Weekly Report #34 (06/19/2018) -Dapp built on Nebulas wins Beijing hackathon
- Weekly Report #33 (06/12/2018) -NIP gets upgraded with Super Contributors
- Weekly Report #32 (06/05/2018) -NAS Nano update features a built-in Dapp Store
- Weekly Report #31 (05/29/2018) -NAS Nano, the official Nebulas mobile wallet, launches on Android and iOS
- Weekly Report #30 (05/22/2018) -the winners for Week 1 of NIP and awarded nearly $350,000 USD in NAS
- Weekly Report #29 (05/15/2018) -Nebulas Attends Blockchain Technology Forum at Google NY
- Weekly Report #28 (05/08/2018) -Our three co-founders attended the Nebulas Community Meeting in Shanghai
- Weekly Report #27 (05/01/2018) -The Nebulas Incentive Program Is About to Kick Off
- Weekly Report #26 (04/24/2018) -NVM new feature design
- Weekly Report #25 (04/17/2018) -Important Announcement on Nebulas Lock Up Bonus Programm
- Weekly Report #24 (04/10/2018) -Dive into NebulasâĂŁâĂŤâĂŁour new Tech column on Medium
- Weekly Report #23 (04/03/2018) -Nebulas Wins Best Performance Award in Innovative District

- Weekly Report #22 (03/27/2018) -Nebulas held an online Tech Reddit AMA
- Weekly Report #21 (03/20/2018) -Nebulas set the launch date for its Mainnet 1.0
- Weekly Report #20 (03/13/2018) -Nebulas held a NYAI Meetup
- Weekly Report #19 (03/06/2018) -Nebulas Global Tour officially kicked off
- Weekly Report #18 (02/27/2018) -Nebulas First Reddit AMA Comes to a Successful Conclusion
- Weekly Report #17 (02/20/2018) -Nebulas Writing Contest Rounded Off
- Weekly Report #16 (02/13/2018) -Nebulas is Holding an Online Reddit AMA
- Weekly Report #15 (02/06/2018) -NebulasâĂŹ Silicon Valley Meetup
- Weekly Report #14 (01/30/2018) -NebulasâĂŹ Trip to Silicon Valley
- Weekly Report #13 (01/23/2018) -Nebulas Davos and Silicon Valley Trips
- Weekly Report #12 (01/16/2018) -Hitters Present at the Blockchain Meetup of The Economist China Readers Club
- Weekly Report #11 (01/09/2018) -Nebulas Testnet Upgraded
- Weekly Report #10 (12/26/2017) -Nebulas CTO Robin Zhong Present at CIE Seminar
- Weekly Report #09 (12/19/2017) -Tsinghua University Talks Well-received
- Weekly Report #08 (12/12/2017) -NAS Token Exchange With Bonus Program a Complete Success
- Weekly Report #07 (12/05/2017) -Nebulas Token Exchange Program with Bonus is ending soon!
- Weekly Report #06 (11/27/2017) -Initiation of âĂIJNAS Token Bonus ProgramâĂİ
- Weekly Report #05 (11/20/2017) -Singapore FinTech Festival
- Weekly Report #04 (11/13/2017) -Columbia University, New York / Nebulas Meetup
- Weekly Report #03 (11/6/2017) -Developing v0.3.0 and improving the Go-nebulas
- Weekly Report #02 (10/30/2017) -Singapore Fintech Festival
- Weekly Report #01 (10/16/2017) -Welcome to the #1 of Nebulas Weekly Report

Ask Me Anything

Nebulas Reddit AMA

- Reddit Questions and Answers - Reddit Weekly Question Recap! (10.29–11.4)
- Reddit Questions and Answers - Reddit Weekly Discussion Recap!ïijĹ10.21–10.26ïijĽ
- Nebulas Reddit AMA Recap - With Nebulas Founder Hitters Xu and Co-founder Aero Wang

- Reddit Questions and Answers - Nebulas Weekly AMA & Constructive Suggestion-sâĂŁâĂŤâĂŁAugust 6 to August 19
- Reddit Questions and Answers - Nebulas Weekly AMA & Constructive Suggestion-sâĂŁâĂŤâĂŁJuly 30 to August 6 2018
- Reddit Questions and Answers - Nebulas Weekly AMA & Constructive Suggestion-sâĂŁâĂŤâĂŁJuly 20 to July 29
- Nebulas First Live Reddit AMA - With Nebulas Founder Hitters Xu
- NebulasâĂŹ First Reddit AMA Recap - Answers and Viewpoints of Nebulas Founder Hitters Xu
- Tech Reddit AMA - With Nebulas CTO Robin Zhong
- Nebulas AMA Series#1 - Testnet with Nebulas Co-Founder and CTO Robin Zhong
- Nebulas AMA Series#2 - Testnet with Nebulas Co-Founder and CTO Robin Zhong
- Nebulas AMA Series#3 - General Question with Nebulas Co-Founder and CTO Robin Zhong
- Answers from AMA - With Nebulas lead core developer Roy Shang

PCTA Reddit AMA Series

- PCTA Reddit AMA Series 1 Recap - Nebulas & XMAX Reddit AMA Recap#Part 1
- PCTA Reddit AMA Series 1 Recap - Nebulas & XMAX Reddit AMA Recap#Part 2
- PCTA Reddit AMA Series 2 Recap - BCH Hard Fork, Beneficial or Harmful
- PCTA Reddit AMA Series 3 Recap - What can we learn from the recent market crash?

Entrevistas da Nebulas

Entrevistas com a Equipa da Nebulas

- Entrevista com a Equipa da Nebulas, SÃl'rie 3 - A Nebulas Pela Qual Estou Ansioso .
- Entrevista com a Equipa da Nebulas, SÃl'rie 2 - PorquÃł Colaborar com a Nebulas .
- Entrevista com a Equipa da Nebulas, SÃl'rie 1 - OpiniÃţes de Nebulers Sobre o Futuro da Blockchain .

Nebulas NOVA, Para Descobrir o Valor dos Dados no Mundo da Blockchain

- Nebulas NOVA, Para Descobrir o Valor dos Dados no Mundo da Blockchain [.(https://www.youtube.com/watch?v=jLIYkG35Ljo)

Entrevistas com Membros do Instituto de InvestigaÃğÃčo da Nebulas

- Entrevista com o LÃ■der do Instituto de InvestigaÃğÃčo da Nebulas, o Dr. Xuepeng Fan - Tomar as RÃľdeas para a CriaÃğÃčo do Instituto de InvestigaÃğÃčo da Nebulas
- Entrevista com o Desenvolvedor PrincÃ■pal da Mainnet da Nebulas, o Dr. Congming Chen - Deixe a Nebulas Voar Mais Alto e Mais Longe!
- Entrevista com o Investigador SÃľnior da Nebulas, o Dr. Zaiyang Tang - O Meu CoraÃğÃčo Pertence Ãă Nebulas, Espero que Brilhemos Juntos
- Entrevista com o Director TÃľcnico da Nebulas, o Dr. Joel - Entrevista Exclusiva com o Director TÃľcnico da Nebulas, o Dr. Joel
- Entrevista com o Investigador SÃľnior do Instituto de InvestigaÃğÃčo da Nebulas, o Dr. Yulong Zeng - O Meu Primeiro Trabalho na Nebulas
- Entrevista com o EstagiÃąrio do Instituto de InvestigaÃğÃčo, o Dr. Dai - A Vida Ãľ um Desafio

Entrevistas com Membros do ComitÃľ TÃľcnico da Nebulas

- Entrevista com o CEO e Fundador da Nebulas, Hitters Xu - Ver Para AlÃľm da Bolha da Blockchain: Sente-se Para Uma Entrevista Com o Fundador da Nebulas.io Hitters Xu

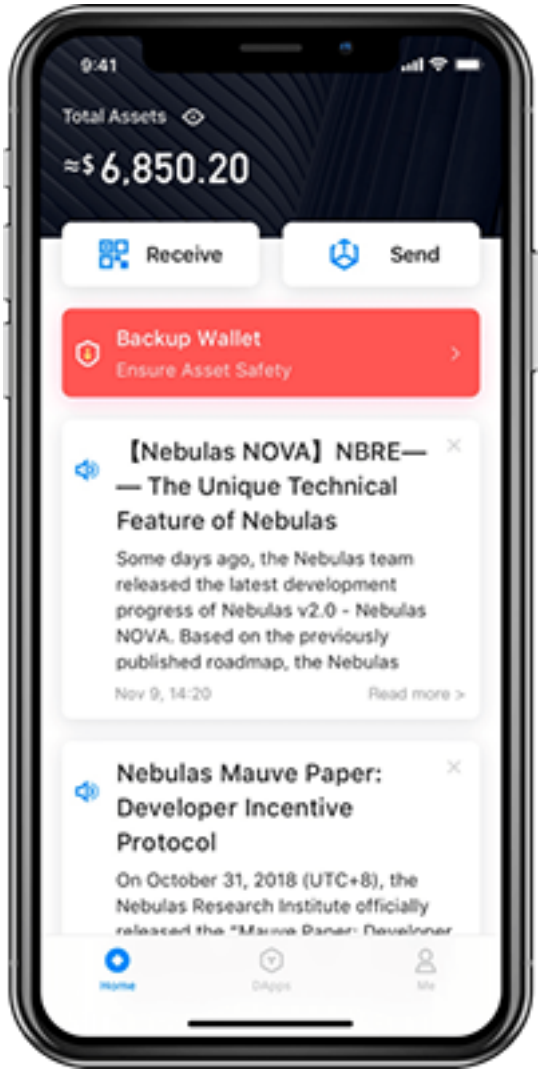
Entrevistas com Membros da Comunidade

- Entrevista com Pluto e Xuxue (CampeÃţes da Semana 1 do Programa de Incentivos da Nebulas) - Programa de Incentivos da NebulasâĂŁâĂŤâĂŁEntrevista com os CampeÃţes da Semana 1
- Entrevista com Jason Mansfield (Vencedor MÃžltiplas vezes da Primeira Temporada do Programa de Incentivos da Nebulas) - Entrevista com Desenvolvedor de DApps na Nebulas: Jason Mansfield
- Entrevista com Honey Thakuria (Vencedor da Accenture Hackathon, usando Nebulas) - Desenvolvimento de DApps e Planeamento de ArquiteturaâĂŁâĂŤâĂŁEntrevista com Honey Thakuria

2.4.10 Ecosystem

NAS Nano

NAS Nano Ãľ a carteira oficial da Nebulas. Pode-a obter aqui.



Nebulas Web Wallet Tutorial

- Part 1 - Criar uma Carteira de NAS
- Part 2 - Envio de NAS da sua Carteira
- Part 3 - Assinar uma TransacÃğÃčo Offline
- Part 4 - Ver InformaÃğÃčo de Carteiras
- Part 5 - VerificaÃğÃčo do Estado de uma TransacÃğÃčo
- Part 6 - ImplementaÃğÃčo de um Smart Contract
- Part 7 - InvocaÃğÃčo de um Smart Contract pela Carteira da Nebulas

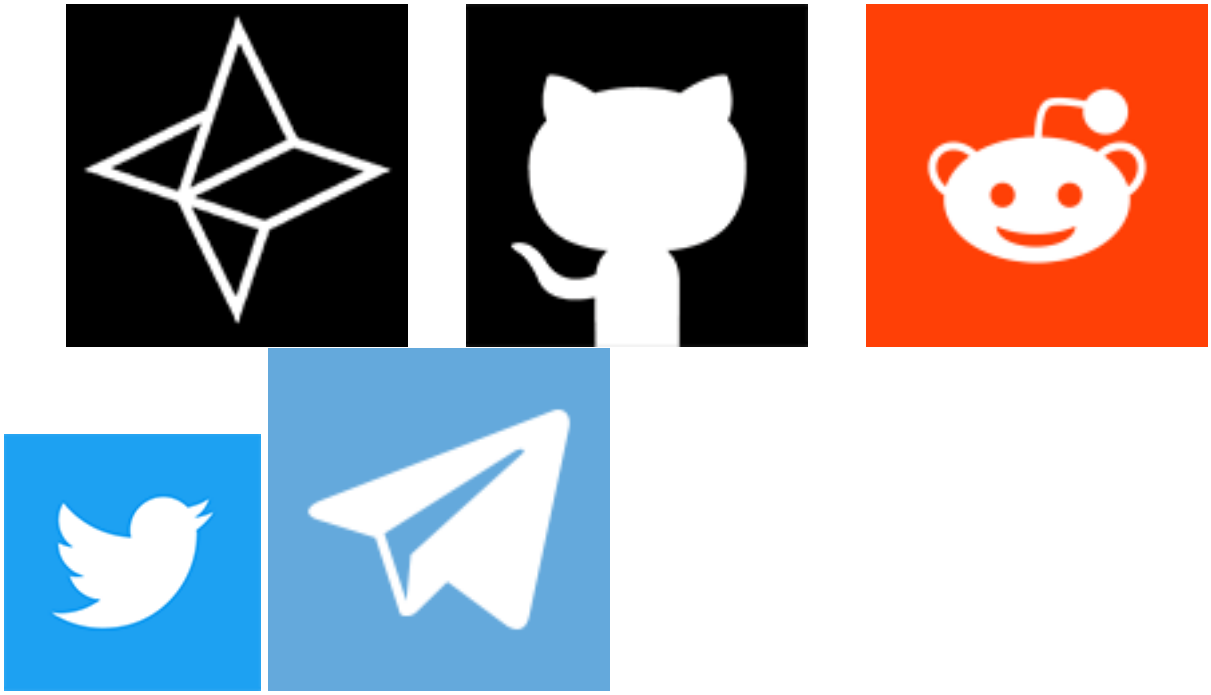
Lista de DApps do Ecossistema

Pode encontrar DApps recomendadas e campeÃčs mensais/semanais da 1Âł Temporada do Programa de Incentivos da Nebulas aqui: SumÃąrio

Convidamo-lo a adicionar mais Ãă lista!

Loja de DApps da Nebulas da comunidade.

2.4.11 Links ÃŽteis



2.4.12 Frequently Asked Questions

Este documento irÃą focar-se na tecnologia por trÃąs da plataforma da Nebulas. Para perguntas mais gerais, por favor veja o FAQ do Reddit.

Para uma compreensÃčo melhor da plataforma da Nebulas, o Livro Branco TÃl'cnico da Nebulas Ãl' recomendado.

Ã■ndice

(d) Smart Contracts

i. Language Support

ii. Ethereum Compatibility

Nebulas Rank (NR)

Mede o valor ao considerar liquidez e propagaÃğÃčo de um endereÃğo. O Nebulas Ranking tenta estabelecer uma abordagem fidedÃ■gna, computacional, e determinÃ■stica. Com o sistema de classificaÃğÃčo de valor, iremos ver mais dApps excepcionais na plataforma da Nebulas.

When will Nebulas Rank (NR) be ready?

O Nebulas Rank foi lanÃğado em Dezembro de 2018. No momento da escrita deste artigo, o servidor NR Query estÃą offline, desde a actualizaÃğÃčo do algoritmo. Pode contribuir para o projecto de refatoraÃğÃčo do cÃşdigo aqui.

Will dApps with more transactions naturally be ranked higher?

NÃčo necessariamente, visto que o nÃžmero de transacÃğÃţes apenas aumenta o grau de in-and-out durante um perÃ■odo de tempo, atÃl' um valor predeterminado. A maneira em que o Nebulas Rank Ãl' calculado usa, entre vÃąrias outras variaveis, a median account stake. A median account stake Ãl' o balanÃğo mediano de um endereÃğo durante um certo perÃ■odo de tempo.

How does the Nebulas Rank (NR) separate quality dApps from highly transacted dApps?

Ao utilizar o Median Account Stake para calcular o NR, o Nebulas Rank garante justiÃğa e resiste manipulaÃğÃčo a um grau elevado, garantindo a probabilidade de dApps de alta qualidade flutuarem para o topo da hierarquia.

Is the Nebulas Ranking algorithm open-source?

Sim.

Who can contribute to the algorithm?

De momento, a equipa da Nebulas estÃą responsÃąvel pelo desenvolvimento do algoritmo. No entanto, todos podem fazer sugestÃţes, relatÃşrios de bugs, e contribuiÃğÃţes de cÃşdigo. O repositÃşrio do SDK encontra-se aqui, e o do Nebulas Rank Offline Service aqui.

Can the Nebulas Rank (NR) algorithm be cheated?

Nada Ãl' invulnerÃąvel, mas o objectivo Ãl' tornar a manipulaÃğÃčo do algoritmo bastante cara e o mais difÃ■cil possÃ■vel.

Nebulas Force (NF)

Suporte a actualizaÃğÃčo dos protocolos centrais e smart contracts nas blockchains. Confere a habilidade de auto-evoluÃğÃčo ao sÃ■stema da Nebulas e Ãăs suas aplicaÃğÃţes. Com o Nebulas Force, desenvolvedores podem criar vÃąrias iteraÃğÃţes de aplicaÃğÃţes complexas, e essas mesmas podem-se adaptar dinamicamente Ãă comunidade ou variaÃğÃţes do mercado.

When will Nebulas Force (NF) be ready?

De acordo com o roadmap o Nebulas Force serÃą lanÃğado no quarto trimestre de 2019.

Can smart contracts be upgraded?

Yes, [short summary explaining how it works]

How is Nebulas Force (NF) smart contract upgrading better than other solutions that are currently or soon-to-be available?

answer here

Can the Nebulas blockchain protocol code be upgraded without forking?

Yes, [short summary explaining how it works]

Can the Nebulas Virtual Machine (NVM) be upgraded?

Yes, [short summary explaining how it works]

Developer Incentive Protocol (DIP)

Designed to build the blockchain ecosystem in a better way. The Nebulas token incentives will help top developers to create more values in Nebulas.

When will the Developer Incentive Protocol (DIP) be ready?

answer here

Will there be a limit as to how many rewards one dApp can receive?

answer here

Will developers still be able to do their own ICOs?

answer here

Will only the top Nebulas Rank (NR) dApps receive rewards?

answer here

How often will rewards be given?

answer here

How will you stop cheaters?

The way the DIP is is designed makes it very hard for cheaters to be successful. Since smart contracts can only be called passively, it would be highly cost ineffective for a user to try to cheat the system. More about this topic can be read in the Technical Whitepaper.

Proof of Devotion (PoD) Consensus Algorithm

To build a healthy ecosystem, Nebulas proposes three key points for consensus algorithm: speediness, irreversibility and fairness. By adopting the advantages of PoS and PoI, and leveraging NR, PoD will take the lead in consensus algorithms.

When will the Proof of Devotion (PoD) Consensus Algorithm be ready?

answer here

What consensus algorithm will be used until PoD is ready?

answer here

How are bookkeepers chosen?

The PoD consensus algorithm uses the Nebulas Rank (NR) to qualify nodes to be eligible. One node from the set is randomly chosen to propose the new block and the rest will become the validators.

Do bookkeepers still have to stake?

Yes, once chosen to be a validator for a new block, the validator will need to place a deposit to continue.

How many validators will there be in each set?

answer here

What anti-cheating mechanisms are there?

answer here

Nebulas Search Engine

Nebulas constructs a search engine for decentralized applications based on Nebulas value ranking. Using this engine, users can easily find desired decentralized applications from the massive market.

When will the Nebulas Search Engine be ready?

answer here

Will you be able to search dApps not on the Nebulas platform?

answer here

Will the Nebulas Search Engine also be decentralized?

answer here

Will the Nebulas Rank (NR) control the search results ranking?

answer here

What data will you be able to search?

We plan many different ways to be able to search the blockchain:

- crawl relevant webpages and establish mapping between them and the smart contracts
- analyze the code of open-source smart contracts
- establish contract standards that enable easier searching

Fundamentals

Nebulas Name Service (NNS)

By using smart contracts, the Nebulas development team will implement a DNS-like domain system named Nebulas Name Service (NNS) on the chain while ensuring that it is unrestricted, free and open. Any third-party developers can implement their own domain name resolution services independently or based on NNS.

When will the Nebulas Name Service be ready?

answer here

When a name is bid on, how long do others have to place their bid?

answer here

How do others get notified that a name is being bid on?

answer here

When a name is reserved who gets the bid amount?

answer here

If I want to renew my name after one year will I need to deposit more NAS?

answer here

Will we be able to reserve names prior to the launch of NNS?

answer here

Lightning Network

Nebulas implements the lightning network as the infrastructure of blockchains and offers flexible design. Any third-party developers can use the basic service of lightning network to develop applications for frequent transaction scenarios on Nebulas. In addition, Nebulas will launch the worldâĂŹs first wallet app that supports the lightning network.

When will lightning network be supported?

answer here

The Nebulas Token (NAS)

The Nebulas network has its own built-in token, NAS. NAS plays two roles in the network. First, as the original money in the network, NAS provides asset liquidity among users, and functions as the incentive token for PoD bookkeepers and DIP. Second, NAS will be charged as the calculation fee for running smart contracts. The minimum unit of NAS is 10âĹŠ18 NAS.

What will happen to the Nebulas ERC20 tokens when NAS is launched?

answer here

Will dApps on the Nebulas platform be able to issue their owns ICOs and tokens?

answer here

Smart Contracts

What languages will be supported when Main-net launches?

answer here

Will Ethereum Smart Contracts (Solidity) be fully supported?

answer here

What other language support will follow (and when)?

answer here

binary storage

What is recommended way to store binary data in Nebulas blockchain? Is it possible at all? Do you encourage such use of blockchain? Also, i couldn‘t find information regarding GlobalContractStorage mentioned in docs, what is it?

> Currently binary data can be stored on chain by binary transaction. The limit size of binary is 128k. But we donâĂŹt encourage storing data on the chain because the user might store some illegal data.
>
> `GlobalContractStorage`not currently implemented. It provides support for multiple contract sharing data for the same developer.

ChainID & connect

Can you tell us what the chainID of Mainnet and Testnet is? I have compiled the source code of our nebulas, but not even our test network?

chainID of Nebulas:

- Mainnet: 1
- Testnet: 1001
- private: default 100, users can customize the values.

The network connection:

- Mainnet:
  - source code:master
  - wiki:Mainnet
- Testnet:
  - source code:testnet
  - wiki:Testnet

smart contract deploy

Our smart contract deployment, I think is to submit all contract code directly, is the deployment method like this?

> Yeah, We can deploy the contract code directly, just as it is to release code to the NPM repository, which is very simple and convenient.

smart contract IDE

We don‘t have any other smart contract ides, like solidity‘s “Remix“? Or is there documentation detailing which contract parameters can be obtained? (because I need to implement

the random number and realize the logic, I calculate the final random number according to the parameters of the network, so I may need some additional network parameters that will not be manipulated.)

> You can use web-wallet to deploy the contract, it has test function to check the parameters and contract execution result.

2.4.13 Licenses

Nebulas Open Source Project License

The preferred license for the Nebulas Open Source Project is the GNU Lesser General Public License Version 3.0 (âĂIJLGPL v3âĂİ), which is commercial friendly, and encourage developers or companies modify and publish their changes.

However, we also aware that big corporations is favoured by other licenses, for example, Apache Software License 2.0 (âĂIJApache v2.0âĂİ), which is more commercial friendly. For the Nebulas Team, we are very glad to see the source code and protocol of Nebulas is widely used both in open source applications and non-open source applications.

In this way, we are still considering the license choice, which kind of license is the best for nebulas ecosystem. We expect to select one of the LGPL v3, the Apache v2.0 or the MIT license. If the latter is chosen, it will come with an amendment allowing it to be used more widely.

Contributor License Agreement

All contributions to Nebulas wikis are licensed under the Creative Commons License SA 4.0.

Colaboradores

Para ver uma lista completa dos nossos colaboradores, clique aqui.

- genindex
- modindex
- search