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

nebulas Documentation

Release 1.0

nebulas

Dec 12, 2018

Category:

CHAPTER 1

Overview

1.1 Welcome to the open-source Nebulas wiki!

Nebulas is the next-generation public blockchain, aiming for a continuously improving ecosystem. Based on its blockchain valuation mechanism, Nebulas proposes future-oriented incentive and consensus systems, and the ability to self-evolve without forking.

Nebulas community is open and everyone can be a contributor and build a decentralized world with us.

The Nebulas wiki is a collaboration tool for the community to publish various documents in a collaborative manner. Include using guides, developing guides, learning resources, and other useful documents.

1.2 Use Wiki

1.2.1 Mainnet

1.2.2 Dapps

1.2.3 Ecosystem

1.2.4 Get Involved

CHAPTER 2

How to Contribute

Your contribution matters!

Nebulas aims for a continuously improving ecosystem, which means we need help from the community. We need your contributions! It is not limited exclusively to programming, but also bug reports and translations, spreading the tenets of Nebulas, answering questions, and so on.

2.1 1. Code & Documentation

2.1.1 1.1. Mainnet Development

We are so excited to devote ourselves to blockchain and to see how blockchain can improve people's lives. We want to share this exciting experience with the whole community. Thus, we call upon developers!

Learn more:

Our githubïijŽgithub.com/nebulasio/go-nebulas

Our Roadmap: nebulas.io/roadmap.html (Stay tuned)

2.1.2 1.2. Bug Reporting

We have always valued bug reporting!

If you find a bug, please report it to the Nebulas community. Bugs may be found on the Nebulas testnet, mainnet, nebPay, neb.js, web wallet, as well as other tools and documen-

tation. We will follow the OWASP Risk Assessment System to calculate the corresponding bounty/reward based on the risk degree of the bug. More TBA.

To submit bugs and related information, please post the information in the related Nebulas mail groups. When submitting reports, please be careful and pay attention to the mail group in order to prevent bugs from being exploited or create duplicates. We welcome you to follow the mail group and join the discussion.

Mail group list: lists.nebulas.io/cgi-bin/mailman/listinfo

Mainnet bug list: lists.nebulas.io/cgi-bin/mailman/listinfo/mainnet-bugs

Testnet bug list: lists.nebulas.io/cgi-bin/mailman/listinfo/testnet-bugs

2.1.3 1.3. Translation

Translating is important to spread Nebulas to the whole world!

We welcome community members from around the world to participate in the translation of Nebulas documentation. You can translate everything from the wiki, including mainnet technical documents, the DApp FAQ, official documents such as the Nebulas White Paper and Yellow Paper, the Nebulas design principle introduction document, and more. Your contribution will significantly help numerous Nebulas developers and community members. Please note that some documents will require a professional background in Math, Computer Science, Cryptography, or other specialties.

2.1.4 1.4. Documentation Writing

Developers in the Nebulas community require documentation to help them understand and use the various functions of Nebulas. The community is welcome and encouraged to write technical introductions and FAQs. In addition, Nebulas community members will also benefit from easy-to-understand introductory guides and user guides on various ecosystem tools.

Your contribution will be of use to all community developers and members, and may also be translated into multiple languages to benefit an even larger amount of members.

2.1.5 1.5. Wiki UI Design

We welcome UI developers to optimize our wiki page and make it more user friendly and easier to read.

Download design template of Nebulas wiki >

Download LOGO material >

If you have any questions or comments, please do not hesitate to post on our github.

2.2 2. User Groups

Communication is key for building a vibrant community. People need to talk with each other in order to share their ideas and thoughts on Nebulas.

Nebulas utilizes several platforms to connect our global community. Please refer to the âĂIJCommunityâĂİ page on official website for more information.

Discord: Available for all community members. You can subscribe to Nebulas News, as well as participate in group discussions. Discord is many users' first choice.

Mailing lists: Discussion groups for core development and bug reporting. We welcome developers to subscribe.

Forum: Reddit/r/nebulas(for all), Reddit/r/nasdev(for developers)

Communication: Slack(for developers), Telegram(for non-developers)

We welcome community developers to create an IRC (Internet Relay Chat) channel for better communication among developers.

2.3 3. Bounties

> *Note:* this is temporary since we believe that bounties are helpful only during the early stages of a community, but can also be harmful in the long-term

We, the Nebulas team, happily introduce several bounties to reward early contributors. Nebulas Bounty rewards include: Developer BountyïijŽplease refer to the Developer Bounty List Wiki Bounty: Based on the contribution, we will give different sized rewards to users who greatly contribute to the Nebulas Wiki, based on Github activity. For example, the reward for translating the Nebulas Yellow Paper will be 200 NAS. The deadline of the first evaluation is Dec. 31, 2018. The specific reward amount and number of recipients will be public. Depending on participation, the reward size will be adjusted in the future.

2.4 4. Donations

We welcome donations to develop Nebulas from the community. Both NAS and ETH are accepted. We also welcome community members to support us in material terms. For example, the donation of meetup locations/spaces, local guides, photography, etcetera. We can also make your contribution known to the community if you would like. If you are an enthusiastic community member and are willing to contribute to our community, please send an email to contact@nebulas.io for more details.

CHAPTER 3

Bug Bounty

3.1 Nebulas Bug Bounty Program

The Nebulas Bug Bounty aims to improve the security of Nebulas Ecosystem, ensuring the establishing for benign Nebulas ecosystem. The Nebulas Bug Bounty Program provides bounties for the discovered vulnerabilities. This bounty program is initiated and implemented by the Nebulas Technical Committee (NTC), joined by the Nebulas technical team and community members. NTC encourages the community to disclose security vulnerabilities via the process described below, and play a role in building Nebulas ecosystem, thereby receiving bounties, and partake in the establishing of Nebulas ecosystem.

3.2 Bug Category

The Bug Bounty Program divides the bug bounties into 2 categories, common bug bounty and special bug bounty. The common bugs include vulnerabilities discovered in Nebulas mainnet, Nebulas testnet, nebPay, Web wallet, neb.js and others, while the special bugs include vulnerabilities found in the inter-contract call function and others.

3.3 Eligibility

The Nebulas Technical Committee will evaluate reward sizes according to the severity calculated by OWASP Risk Rating Method based on **Impact** and **Likelihood**. However, final rewards are determined at the sole discretion of the committee.

Impact:

- High: Bugs affecting asset security.
- Medium: Bugs affecting system stability.
- Low: Other bugs that do not affect asset security and do not affect system stability.

Likelihood:

- High: The bug can be discovered by anyone who performs an operation, regardless of whether or not the bug has been found.
- Medium: Only certain people can discover it (such as a bug that only developers encounter, ordinary users are not affected.)
- Low: Covers less than 1% specific population, such as certain rare Android models; or any other exceptional cases.

3.4 Amount:

To ensure the bug reporter obtains a stable expected reward, the amount in US dollars will be issued in equivalent NAS. The reward amount is divided into 5 categories:

- Critical:US$1,000 or more (No upper limit)
- High: US$500 or more
- Medium: US$250 or more
- Low: US$100 or more
- Improvement: US$30 or more

Note: The Nebulas testnet special vulnerability reward (such as one for testnet inter-contract call function) has been increased accordingly, and the equivalent US dollars are issued in NAS.

3.5 Report A Bug

Please send your bug report via this link.

Notes:

1.Please ensure the accuracy and clarity of the content, because the reward evaluation will be based on the content submitted in this form. 2.If many people discover the same bug, then their report submissions in chronological order will determine their reward. Community users are welcome to discuss the issues of bugs, but the discussion itself is not considered a report, therefore a report form must still be submitted.

3.6 Notes:

1.The Nebulas Bug Bounty Program is long-standing. The Nebulas Technical Committee reserves the right to final interpretation of this program, and the rights to adjust or cancel the reward scope, eligibility and amount. 2.The Nebulas Technical Committee will confirm and evaluate the bug report after its submission. The evaluation time will depend on the severity of the problem and the difficulty of fixing it. The evaluation result will be sent to its reporter by email as soon as possible. 3.To avoid bugs being exploited, reporters are required to submit the bug bounty application to bug report entrance. 4.Reporters shall keep the bugs non-public and confidential until 30 days after submitting the bugs to Nebulas, and shall not disclose the bugs to any third party. Such confidentiality timeline can be extended by Nebulas unilaterally. If reporters disclose the bugs to any third party which cause any harm to Nebulas or NebulasâĂŹ users, reporters shall be responsible of the compensation for all the losses. 5.The Nebulas Technical Committee encourages community member to discuss with the Nebulas technical team and other community members in the Nebulas public discussion group. We also encourage our community members to join us in fixing these bugs. Welcome to join our Nebulas maillist for discussion.

CHAPTER 4

Wiki Using Guide

There will be three parts:

4.1 How to use this wiki

TBA

4.2 Editing pages on Github

You should use reST to edit .rst file, and Pandoc Markdown to edit .md file.

Click here to learn the different between Markdown and reST.

These are some learning resources:

- How to use Markdown by John Gruber
- Markdown Guide by iA Writer

When you edit pages on Github, you can click "Preview changes" to view the results. You can check the building processing here.

4.3 Tracking changes

TBA

CHAPTER 5

Autonomous Metanet

5.1 Nebulas Force (NF)

A series of basic protocols such as the NR, the PoD, and the DIP shall become a part of the blockchain data. With the growth of data on Nebulas, these basic protocols will be upgraded, which will avoid fractures between developers and community, as well as a âĂIJforkâĂİ. We call this fundamental capability of our blockchain âĂIJNebulas ForceâĂİ (NF).

As the Nebulas community grows, NF and basic protocolsâĂŹ update ability shall be open to the community. According to usersâĂŹ NR weight and the community voting mechanism, NebulasâĂŹ evolution direction and its update objectives will be determined by the community. With the help of NFâĂŹs core technology and its openness, Nebulas will have an ever-growing evolutive potential and infinite evolving possibilities.

5.1.1

5.2 Nebulas Incentive (NI)

The Nebulas Incentive includes Proof of Devotion (PoD) and the Developer Incentive Protocol (DIP).

5.2.1 Proof-of-Devotion (PoD)

Based on NebulasâĂŹ NR system, we shall adopt the PoD (Proof-of-Devotion) consensus algorithm. PoD gives an âĂIJinfluentialâĂİ user of the Nebulas Blockchain an opportunity to become a bookkeeper and receive NebulasâĂŹ block rewards and transaction fees as revenues,

which will in turn encourage them to continuously contribute to NebulasâĂŹ stability and security.

5.2.2 Developer Incentive Protocol (DIP)

On Nebulas, we proposed the concept of DIP (Developer Incentive Protocol) for developers of smart contracts and DApps. DIPâĂŹs core concept: in the interval of pre-specified blocks, for those developers whose smart contracts and DApps were deployed online during the most recent interval, with a NR value higher than a specified threshold, DIP shall reward them the corresponding developer incentives (NAS token), and these incentives shall be recorded on blocks by bookkeepers. With the DIPâĂŹs positive incentive mechanism, more and more developers will get incentives to create valuable smart contracts and DApps, which will help to build a positive feedback ecosystem for the developer community.

5.3 Nebulas Rank (NR)

Nebulas Rank (NR) is an open source ranking algorithm used to measure the influence of relationships among addresses, smart contracts, and distributed applications (DApps). It helps users utilize information within the ever-increasing amount of data on all blockchains, but it also helps developers to use our search framework directly in their own applications.

On Nebulas, we measure value regarding:

- **Liquidity**

Finance is essentially the social activities which optimize social resources via capital liquidity and in turn promotes economic development. Blockchains establish a value network in which the financial assets can flow. Daily volume of Bitcoin and Ethereum, which are most familiar to us, already exceeds $1 billion. From this data, we can see that the higher the transaction volume and transaction scale, the higher the liquidity. As a consequence of this, higher liquidity will increase the quantity of transactions and enhance the value. That will further strengthen the value of the financial assets, creating a complete positive feedback mechanism. Therefore liquidity, i.e. transaction frequency and scale, is the first dimension that NR measures.

- **Propagation**

Social platforms like WeChat and Facebook have almost 3 billion active users per month. Social platformsâĂŹ rapid user growth is a result of the reflection of existing social networks and stronger viral growth. In particular, viral transmission, i.e. speed, scope, depth of information transmission and linkage, is the key index to monitor the quality of social networks and user growth. In the blockchain world, we can see the same pattern. Powerful viral propagation indicates scope and depth of asset liquidity, which can promote its asset quality and asset scale. Thus, viral transmission, i.e. scope and depth of asset liquidity, is the second dimension that NR measures.

- **Interoperability**

During the early stages of the internet, there were only basic websites and private information. Now, information on different platforms can be forwarded on the network, and isolated data silos are gradually being broken. This trend is the process of identifying higher dimensional information. From our point of view, the world of blockchains shall follow a similar pattern, but its speed will be higher. The information on usersâĂŹ assets, smart contracts, and DApps will become richer, and the interaction of higher dimensional information shall be more frequent, thus better interoperability shall become more and more important. Therefore, the third dimension measured by the NR is interoperability.

Based on the aforementioned dimensions, we started constructing NebulasâĂŹ NR system by drawing from richer data, building a better model, digging up more diversified value dimensions, and establishing a measure of value in the blockchain world.

5.4 Roadmap of Autonomous Metanet

Please visit our new Roadmap here.

5.4.1 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.

5.4.2 Long term goals

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

5.4.3 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.

CHAPTER 6

Go-Nebulas

6.1 WhatâĂŹs Nebulas

6.1.1 Nebulas: Next Generation Public Blockchain

6.1.2 Nebulas is aiming to build a continuously improving ecosystem.

Nebulas is a next-generation public blockchain. It introduces Nebulas Rank (NR), a new measure of value for every unit of the blockchain universe, like addresses, DApps and smart contracts. Based on NR, it involves Nebulas Incentive (NI), which motivates developers with Developer Incentive Protocol, and users with the Proof of Devotion consensus algorithm. Moreover, it proposes Nebulas Force (NF), which gives the blockchain and smart contracts within it a self-evolving capacity. In unison, NR, NI, and NF produce a continuously improving and expanding blockchain ecosystem.

There are three technical features: value ranking, self-evolution, and native incentive.

Facing the opportunity and challenge as above, we aim to create a self-evolving blockchain system based on value incentive.

6.1.3 Principles

The Nebulas blockchain has three major principles:

Nebulas Rank (NR)

Nebulas Rank (NR) is an open source ranking algorithm used to measure the influence of relationships among addresses, smart contracts, and distributed applications (DApps). It helps users utilize information within the ever-increasing amount of data on all blockchains, but it also helps developers to use our search framework directly in their own applications.

On Nebulas, we measure value regarding:

- **Liquidity**

Finance is essentially the social activities which optimize social resources via capital liquidity and in turn promotes economic development. Blockchains establish a value network in which the financial assets can flow. Daily volume of Bitcoin and Ethereum, which are most familiar to us, already exceeds $1 billion. From this data, we can see that the higher the transaction volume and transaction scale, the higher the liquidity. As a consequence of this, higher liquidity will increase the quantity of transactions and enhance the value. That will further strengthen the value of the financial assets, creating a complete positive feedback mechanism. Therefore liquidity, i.e. transaction frequency and scale, is the first dimension that NR measures.

- **Propagation**

Social platforms like WeChat and Facebook have almost 3 billion active users per month. Social platformsâĂŹ rapid user growth is a result of the reflection of existing social networks and stronger viral growth. In particular, viral transmission, i.e. speed, scope, depth of information transmission and linkage, is the key index to monitor the quality of social networks and user growth. In the blockchain world, we can see the same pattern. Powerful viral propagation indicates scope and depth of asset liquidity, which can promote its asset quality and asset scale. Thus, viral transmission, i.e. scope and depth of asset liquidity, is the second dimension that NR measures.

- **Interoperability**

During the early stages of the internet, there were only basic websites and private information. Now, information on different platforms can be forwarded on the network, and isolated data silos are gradually being broken. This trend is the process of identifying higher dimensional information. From our point of view, the world of blockchains shall follow a similar pattern, but its speed will be higher. The information on usersâĂŹ assets, smart contracts, and DApps will become richer, and the interaction of higher dimensional information shall be more frequent, thus better interoperability shall become more and more important. Therefore, the third dimension measured by the NR is interoperability.

Based on the aforementioned dimensions, we started constructing NebulasâĂŹ NR system by drawing from richer data, building a better model, digging up more diversified value dimensions, and establishing a measure of value in the blockchain world.

Nebulas Force (NF)

A series of basic protocols such as the NR, the PoD, and the DIP shall become a part of the blockchain data. With the growth of data on Nebulas, these basic protocols will be upgraded,

which will avoid fractures between developers and community, as well as a âĂIJforkâĂİ. We call this fundamental capability of our blockchain âĂIJNebulas ForceâĂİ (NF).

As the Nebulas community grows, NF and basic protocolsâĂŹ update ability shall be open to the community. According to usersâĂŹ NR weight and the community voting mechanism, NebulasâĂŹ evolution direction and its update objectives will be determined by the community. With the help of NFâĂŹs core technology and its openness, Nebulas will have an ever-growing evolutive potential and infinite evolving possibilities.

Nebulas Incentive (NI)

The Nebulas Incentive includes Proof of Devotion (PoD) and the Developer Incentive Protocol (DIP).

Proof-of-Devotion (PoD)

Based on NebulasâĂŹ NR system, we shall adopt the PoD (Proof-of-Devotion) consensus algorithm. PoD gives an âĂIJinfluentialâĂİ user of the Nebulas Blockchain an opportunity to become a bookkeeper and receive NebulasâĂŹ block rewards and transaction fees as revenues, which will in turn encourage them to continuously contribute to NebulasâĂŹ stability and security.

Developer Incentive Protocol (DIP)

On Nebulas, we proposed the concept of DIP (Developer Incentive Protocol) for developers of smart contracts and DApps. DIPâĂŹs core concept: in the interval of pre-specified blocks, for those developers whose smart contracts and DApps were deployed online during the most recent interval, with a NR value higher than a specified threshold, DIP shall reward them the corresponding developer incentives (NAS token), and these incentives shall be recorded on blocks by bookkeepers. With the DIPâĂŹs positive incentive mechanism, more and more developers will get incentives to create valuable smart contracts and DApps, which will help to build a positive feedback ecosystem for the developer community.

Value ranking

To enable value discovery in blockchain, **Nebulas Rank** measures multidimensional data in the blockchain world and powers the decentralized search framework.

Self-evolution

To avoid the damage caused by forking to the blockchain, **Nebulas Force** enables rapid iteration and upgradability to its blockchain without the need for hard forks.

Native incentives

With forward-looking incentive and consensus mechanisms, the **Nebulas Incentive** rewards developers and users who contribute to the sustainability and growth of the ecosystem.

This is an excerpt of the Nebulas Non-technical Whitepaper.

If you want to know more about Nebulas, please subscribe to the official blog, or visit our website: nebulas.io. Read our Non-technical White Paper (English), Technical White Paper (English).

Thank you.

6.2 Nebulas Technical Committee

6.2.1 Our Goal

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.

Learn more about Nebulas Technical Committee, please visit Nebulas website.

6.2.2 Meeting Mins

Meeting Mins 2: Nebulas Nova Tech Tradeoffs Nov, 29, 2018

Meeting Mins 1: Nebulas Nova Tech Tradeoffs Nov, 21, 2018

6.3 Papers

6.3.1 Nebulas White Paper

6.3.2 Nebulas Technical White Paper

You can download PDF here.

Welcome to translate Yellow Paper on Nebulas wiki. Learn more about how to contribute.

6.3.3 Nebulas Rank Yellow Paper

You can download PDF here.

Welcome to translate Yellow Paper on Nebulas wiki. Learn more about how to contribute.

6.3.4 Developer Incentive Protocol Mauve Paper

You can download PDF here.

Welcome to translate Mauve Paper on Github, and you can report bugs on Github too. View here.

Learn more about how to contribute.

6.4 How to Develop

6.4.1 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.

6.4.2 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£ŹçŕĞç§■æŰĞå§žäžŐMac OSX åŠŇ Ubuntuçşżçż§ïijŇçőĂå■ŢäżŃçż■åęĆä¡ŢèřČèŕŢGo-NebulaséąźçŻőïijŇäÿżèęĄäżŃçż■äÿĽ'çğ■æŰźæşŢèřČèŕŢïijŽdlvåŚ¡äžd'èąŇèřČèŕŢïijŇGogland IDEèřČèŕŢïijŇäžěåŔŁVisual Studio CodeèřČèŕŢãĂĆ

èřČèŕŢåŹÍDelveåőĽèčĚ

åIJÍ Mac OSX äÿŁåőĽèčĚDelve

GoogleåőŸæŰźäÿžgolangçŽĎèřČèŕŢä¿Ńå■ŘçŤÍgdbïijŇä¡ĘæŸŕdelveæŸŕæŻťåŘĹéĂĆçŽĎèřČèŕŢåŹĺïij
NebulasãĂĆæŹőéĂŽçŽĎgoéąźçŻőæŸŕåŔŕäžěçŽĎïijŇåĚůä¡Şä¡ŞçŐřåřśæŸŕèřČèŕŢGo-
Nebulaséą́źçŻőæŮűïijŇæŰ■çĆźæŮăæşŢåĄIJä¡ŔïijŇäijŽæřÿè£IJhangä¡ŔãĂĆæĹŚäžňå£ĚéążäžŐgithubäÿŁäÿ
binaryïijŇæ■ěéłď åęĆäÿŃïijŽ

åĚĹçŤÍHomebrewåőĽèčĚæIJL'bugçŽĎDelveïijŽ

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

åőL'èčĚæ■ď æIJL'éŮőécŸçŽĎDelveïijŇåĚűåőđåřśæŸŕäÿžäžĘèőľ åőČåÿőæĹŚäžňåIJÍMacæIJŽåŹĺäÿŁç■¿
certèŕĄäźęãĂĆåęĆæđIJä¡äèĞłåůśæĎ£æĎŔçźĄçŘŘçŽĎæL'ŃåŁĺåĹŻåžžèŕĄäźęïijŇäź§åŔŕäžěäÿ■çŤÍåőL'èčĚ
//github.com/derekparker/delve/blob/v0.12.2/Documentation/installation/
osx/install.mdçŽĎãĂŘCreate a self-signed certificateãĂŚãĂĆ
çňňäžŇæİąrmåŚ¡äžď æŸŕäÿžäžĘåĹäéŹď è£ŹäÿłæIJL'éŮőécŸçŽĎdlv bina-
ryïijŇæĹŚäžňéIJÄèęĄäžŐæžŘçăĄçijŰèŕŚåĞžäÿĂäÿłæ■čçąőçŽĎçL'ĹæIJňïijŇåźűäÿŤåĹ'çŤÍHomebewäÿžæĹ
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
```

åĹŻåžžäÿĂäÿłäÿťæŮűæŰĞäžűåď źïijŇäžŐgithubäÿŃè¡¡äžčçăĄãĂĆæşĺæĎŔæŰĞäžűåď 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¡çŽĎdlvæŃůèť İè£Ż/usr/local/b
debuggerãĂĆè¿ŞåĚěåŚ¡äžď dlv versionïijŇåęĆæđIJèČ¡æ■čåÿÿè£ŘèąŇïijŇæŸ¿çď žçL'ĹæIJňåŘůïijŇèŕť æŸŐ

åIJÍ Ubuntu äÿŁåőĽèčĚDelve

åŕźäžŐUbuntuçşžçž§ïijŇåŔŕäžěçŻť æŐěä¡£çŤÍäÿŃéİćçŽĎæŇĞäžď 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¡¡äžčçăĄąĂĆæşĺæĎŕæŰĞäžűåď źäÿ■æăĞæşĺçžč

DelveåŚ¡äžď'èąŇèřČèŕŢ åęĆæđIJä¡ăäžěåL'■çŤĺgdbèřČèŕŢè£ĞCçĺŃåžŕïijŇåŕźdlvåŚ¡äžď'èąŇèřČèŕŢçŽ
//github.com/derekparker/delve/blob/master/Documentation/usage/dlv.md
è£ŹéĞŇåŕłäžŃçż■debugéČĺåĹĘãĂĆ

è¿ŞåĚěåęĆäÿŃåŚ¡äžď'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¡äžď'

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

dlvèřČèŕŢåŹĺæŔŘçď'žäžčçăĄářĘåIJĺcmd/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§ěçIJŃåŔŸéĞŕïijŇåŕźçŤĺprintåŚ¡äžď'ïijŽ

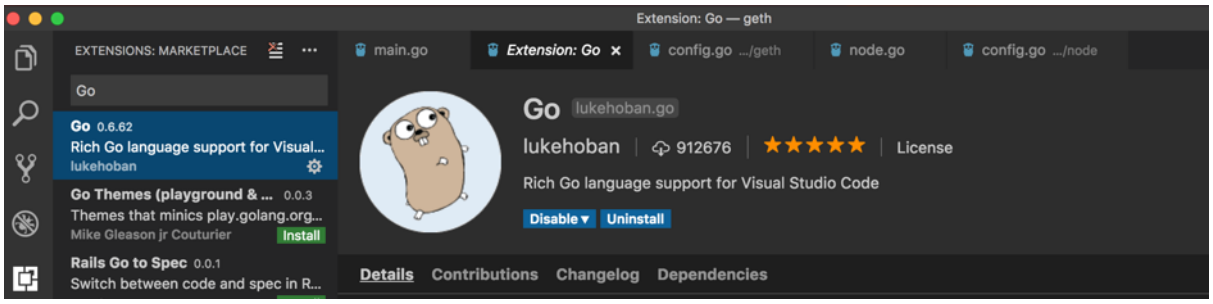
```
(dlv) print ctx
*github.com/nebulasio/go-nebulas/vendor/github.com/urfave/cli.
↪Context {
```

```
    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),
```

æŻť'åď'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¡ŕåĚňåŘÿåŔŚåÿČçŽĎèůĺáźşåŘŕäžčçăAçijŰè¿ŚåůěåĚůïijŇäÿŃè¡¡åIJŕåİĂïijŽ //code.visualstudio.com/Download VS CodeéIJĂèęĄåőL'èčĚGoæŔŠäžű



æĽŞåijĂæŰĞäžűåď'ź/Users/xxx/workspace/blockchain/src/github.com/nebulasio/go-nebulas/ïijŇåIJĺ.vscodeæŰĞäžűåď'źäÿŃåĹŻåžžäÿď'äÿłæŰĞäžűsettings.jsonåŠŇlaunch.jsonãĂĆsettings.jsonæŰĞäžűåĘĚåőźïijŽ

```
// Place your settings in this file to overwrite default and user␣
↪settings.
{
    "go.gopath": "/Users/xxx/workspace/blockchain/",
    "go.formatOnSave": true,
```

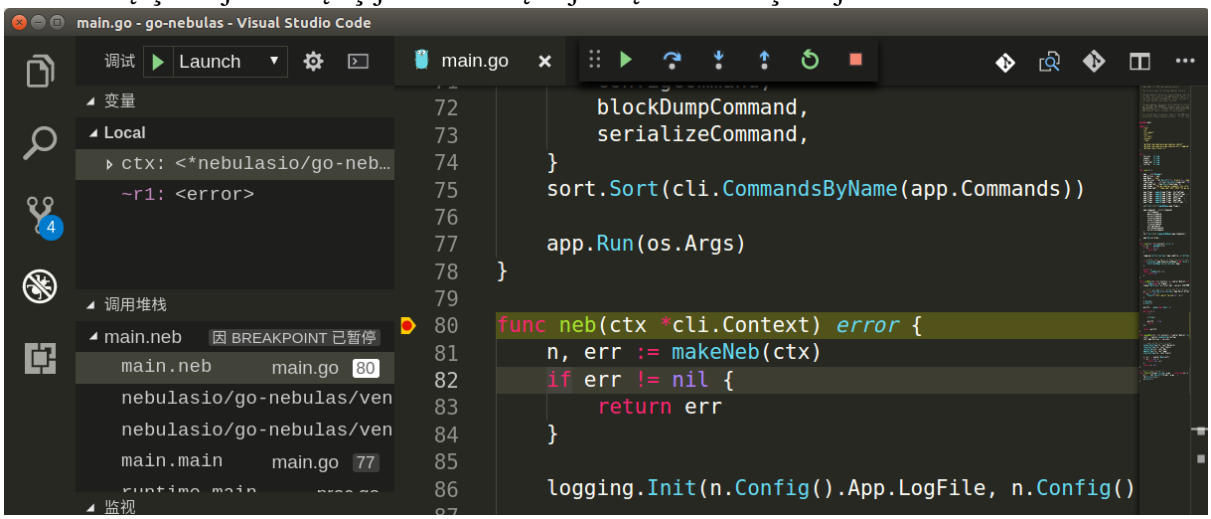
```
    "go.gocodeAutoBuild": false,
    "go.toolsGopath": "/Users/xxx/workspace/gotools",
    "explorer.openEditors.visible": 0,
}
```

go.toolsGopathæŸŕanalysis toolsåőL'èčÉçŽĎåIJřåİĂïijŇåŔŕäžěæŇĞåőŽäÿžäżżä¡ŢçŽőå¡ŢïijŇè£ŹäžŻantoolsåŔŕäžěä¿ŻåĚűåőČworkspaceåĚšäžńãĂĆ

launch.jsonæŰĞäżűåĘĚåőźïijŽ

```
{
    "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
        }
    ]
}
```

åIJÍcmd/neb/main.goïijŇnebåĞ¡æŢřäÿ■èő¿ç¡őæŰ■çĆźïijŇF5è£ŘèąŇïijŇGo-NebulaséąźçŻőäijŽè£ŻèąŇçijŰèŕŚè£ŘèąŇïijŇåĄIJåIJÍæŰ■çĆźïijŽ



çĎűåŘŐïijŇåřśåŔŕäžěåijĂå£ČçŽĎåŘŕåŁÍNebulasäžčçăĄèřČèŕŢäźŃæŮĚïijĄ

6.4.3 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

6.4.4 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

6.5 Todo List

6.5.1 Go-Nebulas

- åÿőåŁľæţŃèŕŢNebulasïijŇèőľNebulasæŻť'åŁăåĄěåčőïijŇæŻť'åď'ŽèŕęæČĚ
- åÿőåŁľNebulasåijĂåŔŚåď'ŽèŕěĺĂçŽĎsdkåžŞïijŇåęĆNode, Ruby, Python, Php, Javaç■Ľ'ç■Ľ'ïijŇæŐěåŔčæŰĞæąč
- åÿőåŁľNebulasåijĂåŔŚåŔŕäžď'äžŠæŰĞæąčïijŇæŐěåŔčæŰĞæąč

6.5.2 Research

- åÿőåŁľäijŸåŇŰNebulas RankçőŮæşŢïijŇæŻť'åď'ŽèŕęæČĚ
- åÿőåŁľNebulasåĹľçŤĺå¡ćåijŔåŇŰèŕĄæŸŐçŽĎæŰźåijŔèŕĄæŸŐåĚśèŕĘçőŮæşŢPoDçŽĎåőĽ'åĚĺæĂğï

6.5.3 Wiki

- åÿőåŁľNebulasæŁŁwikiç£żèŕŚæĹŘä¡ăçŽĎæŕ■èŕ■ïijŇèőľæŻť'åď'ŽäžžäžĘèğčåŇžåİŮéŞ¿ïijŇäžĘèğčNe
- åÿőåŁľNebulasåĹűä¡IJçğŚæŹőæŢŹçĺŃïijŇåÿőåŁľæŻť'åď'ŽæšąæIJĽçijŰçĺŃèČŇæŹŕçŽĎäžžçŘĘèğčåź

6.5.4 Explorer

- åÿőåŁľNebulasåőŇåŰĎåŇžåİŮæţŔèğĹåŹĺïijŇæŻť'åď'ŽèŕęæČĚ

6.5.5 Wallet

- åÿőåŁľNebulasåőŇåŰĎç¡Śéąţç L'ĹéŠśåŇĚïijŇæŻť'åď'ŽèŕęæČĚ
- åÿőåŁľNebulasåőŇåŰĎæąŇéİćçL'ĹéŠśåŇĚïijŇæŻť'åď'ŽèŕęæČĚ

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.

6.6 Roadmap of Nebulas

Please visit our new Roadmap here.

6.6.1 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.

6.6.2 Long term goals

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

6.6.3 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.

6.7 Frequently Asked Questions

This document will focus on the technology behind the Nebulas platform. For broader questions, please view the Reddit FAQ.

For a better understanding of the Nebulas platform it's highly recommended to read the Nebulas Technical Whitepaper.

Table of Contents

6.7.1 Nebulas Rank (NR)

Measures value by considering liquidity and propagation of the address. Nebulas Ranking tries to establish a trustful, computable and deterministic measurement approach. With the value ranking system, we will see more and more outstanding applications surfacing on the Nebulas platform.

When will Nebulas Rank (NR) be ready?

answer here

Will dApps with more transactions naturally be ranked higher?

answer here

How does the Nebulas Rank (NR) separate quality dApps from highly transacted dApps?

answer here

Is the Nebulas Ranking algorithm open-source?

Yes

Who can contribute to the algorithm?

At this time the Nebulas core team is responsible for the development of the algorithm. Over time we will open it up to the community to contribute and vote to determine the future of the algorithm.

Can the Nebulas Rank (NR) algorithm be cheated?

We will implement strict manipulation controls, and of course the Nebulas Rank (NR) will continually be evolving to meet the needs of the community.

6.7.2 Nebulas Force (NF)

Supports upgrading core protocols and smart contracts on the chains. It provides self-evolving capabilities to Nebulas system and its applications. With Nebulas Force, developers can build rich applications in fast iterations, and the applications can dynamically adapt to community or market changes.

When will Nebulas Force (NF) be ready?

answer here

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]

6.7.3 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.

6.7.4 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

6.7.5 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

6.7.6 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.

6.8 Infrastructure

6.8.1 Network Protocol

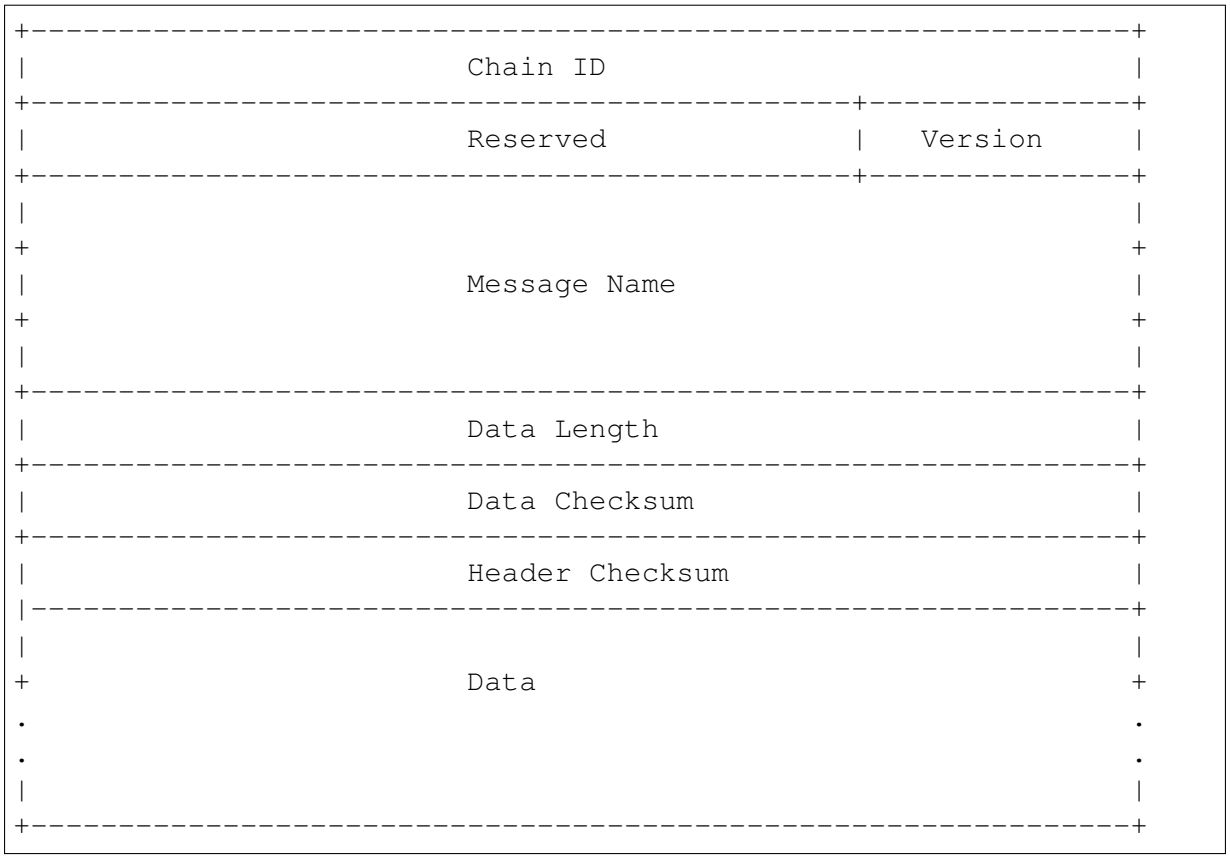
For the network protocol, there were a lot of existing solutions. However, the Nebulas Team decided to define their own wire protocol, and ensure the use of the following principles to design it:

- 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.

6.8.2 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.

6.8.3 Nebulas V8 Engine

Nebulas V8 Engine is

6.8.4 LLVM Engine

TBD.

6.8.5 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.gt(0)) {
        return false;
    }
    return true;
};
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.UNPAYABLE : 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);
    })
```

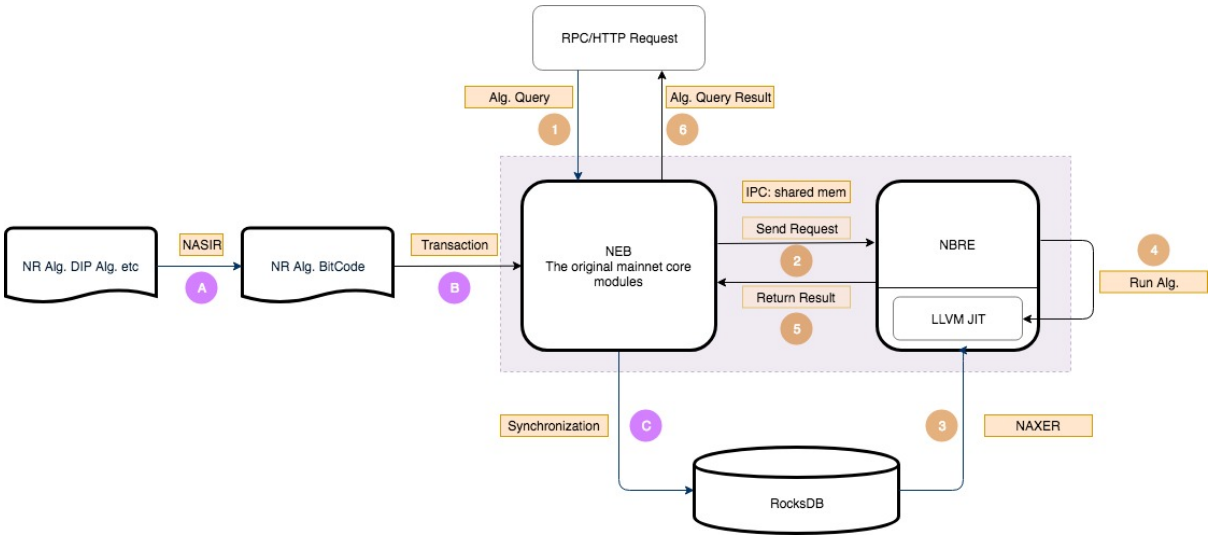
Tips:

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.

6.8.6 NBRE Design Doc

NBRE (Nebulas Runtiome Environment) is the Nebulas chain execution environment. Its framework is shown as follows.



NBRE contains two main processes, which provide the methods how to update algorithms and how to execute algorithms.

The updating process provides how to upload algorithms and core protocols. It includes the following steps:

1. The algorithms are implemented with the languages supported by LLVM. Then, their codes are handled by the NASIR tool, which are translated to bitcode.
2. The bitcode streams are coded with base64, which are translated to payload of transaction data. The transaction data is uploaded to the online chain.
3. After that, the transaction data will be packed and varified. Then, the related bitcode will stored into the RocksDB.

The execution process exhibits the processes from request to results. The corresponding details are as follows.

1. User appries for algorithm call requests with the forms of RPC or RESful API.
2. After receiving the request, the core NEB forward it to NBRE.
3. NBRE starts JIT and loads the algorithm code into JIT.

4. The JIT executes the algorithm with specified parameters and the invoking method, and returns the execution result.
5. NBRE returns the execution result to NEB through IPC.
6. NEB returns the result to the user.

IPC

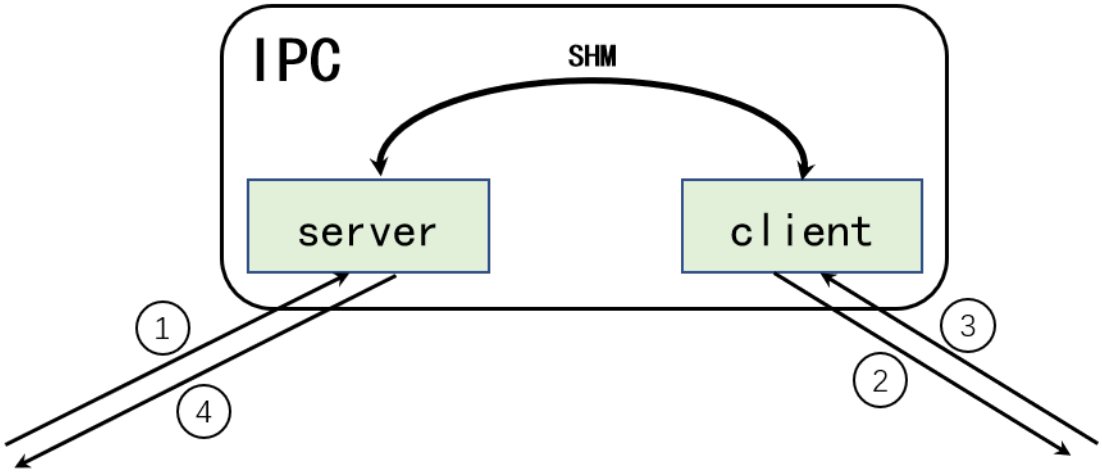
IPC is the messenger for NEB and NBRE interaction.

Features

IPC adopts shared memoty to communicate between NEB and NBRE to improve performance. There are two sub-threads, a server and a client, inside IPC. The server listens for the NEB request, and the client listens for the NBRE result. Also, there is communication interaction between the two threads.

Framework

The framework of IPC is shown as below.



1. NEB calls a function, and the server receives the request and sends it to the client.
2. The client sends the request to NBRE.
3. NBRE runs the corresponding program and returns the result to the client, the client sends the result to the server.
4. The server returns the result to the NEB.

JIT

JIT is a concurrent virtual machine based on LLVM, which runs ir programs providing algorithms and interfaces for NBRE. It is the key of the dynamic update for NBRE.

Features

Dynamic update

The dynamic update in NBRE contains two respects: - NBRE's own dynamic update - NBRE's new feature interfaces

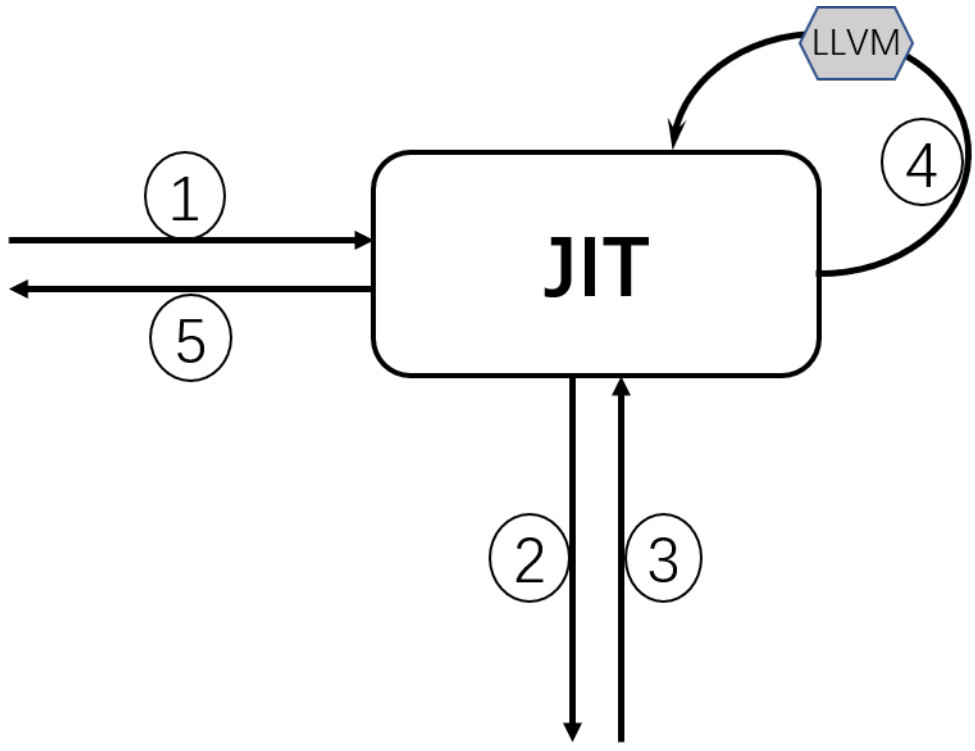
NBRE's updates are performed by adding algorithms and interface programs to the database. When a new function is updated or called, the corresponding program will be loaded into the JIT in the database.

Concurrent virtual machine

To improve performace, JIT is implemented based on a concurrent virtual machine mechanism. When one interface is called, the JIT first queries whether the corresponding program has been loaded. If the programs is loaded, sets its execution count to be 1800; otherwise, loads the program from database and sets its execution count to be 1801. Then runs the corresponding progrm. At regular intervals, the JIT decrements the corresponding count of each loaded function by one and releases the program with a count when its count less than zero.

Framework

The JIT framework is shown as below.



1. One interface is requested from outside.
2. JIT queries the corresponding function program from the database.
3. JIT loads the corresponding program.
4. Runs the program.

5. Returns the result.

CHAPTER 7

DApp Development

7.1 How to Join Nebulas Mainnet

7.1.1 Introduction

We are glad to release Nebulas Mainnet here. Please join and enjoy Nebulas Mainnet.

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

Configuration

The Mainnet configuration files are in folder `mainnet/conf`, including

genesis.conf

All configurable information about genesis block is defined in genesis.conf, including

- **meta.chain_id:** chain identity
- **consensus.dpos.dynasty:** the initial dynasty of validators
- **token_distribution:** the initial allocation of tokens

Attention: DO NOT change the genesis.conf.

config.conf

All configurable information about runtime is defined in config.conf.

Please check the `template.conf` to find more details about the runtime configuration.

Tips: the official seed node info is as below,

```
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"]
```

API List

Main Endpoint:

| API | URL | Protocol | | — | :—: | :—: | | RESTful | https://mainnet.nebulas.io/ | HTTP |

- GetNebState : returns nebulas client info.
- GetAccountState: returns the account balance and nonce.
- Call: execute smart contract local, don't submit on chain.
- SendRawTransaction: submit the signed transaction.
- GetTransactionReceipt: get transaction receipt info by tansaction hash.

More Nebulas APIs at RPC.

7.1.2 Tutorials

English

1. Installation (thanks Victor)
2. Sending a Transaction (thanks Victor)
3. Writing Smart Contract in JavaScript (thanks otto)
4. Introducing Smart Contract Storage (thanks Victor)
5. Interacting with Nebulas by RPC API (thanks Victor)

äÿ■æŰĞ

1. çijŰèŕŠåőL'èčĚåŔŁè£ŘèąŇneb
2. åIJÍæŸ§äžŠéŞ¿äÿŁåŔŠéĂĄäžd'æŸŞ
3. ä¡£çŤÍJavaScriptçijŰåĘŹæŹžèČ¡åŘĹçžę
4. æŹžèČ¡åŘĹçžęå■ŸåĆÍåŇžäżŃçż■
5. éĂŽè£ĞRPCæŐěåŕčäÿŐæŸ§äžŠéŞ¿äžd'äžŠ

7.1.3 Contribution

Feel free to join the Nebulas Mainnet. If you have found something wrong, please submit an issue or submit a pull request to let us know, and we will add your name and url to this page as soon as possible.

7.2 How to Join Nebulas Testnet

7.2.1 Introduction

We are glad to release the Nebulas Testnet. It simulates the Nebulas network and NVM, and allows developers to interact with Nebulas without paying the cost of gas.

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

Configuration

The testnet configuration files are in the folder `testnet/conf` under `testnet` branch, including

genesis.conf

All configurable information about the genesis block is defined in genesis.conf, including

- **meta.chain_id:** chain identity
- **consensus.dpos.dynasty:** the initial dynasty of validators
- **token_distribution:** the initial allocation of tokens

Attention: DO NOT change the genesis.conf.

config.conf

All configurable information about runtime is defined in config.conf.

Please check the `template.conf` to find more details about the runtime configuration.

Tips: the official seed node info is as below,

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

API List

Test Endpoint:

| API | URL | Protocol | | — | :—: | :—: | | RESTful | https://testnet.nebulas.io/ | HTTP |

- GetNebState : returns nebulas client info.
- GetAccountState: returns the account balance and nonce.
- LatestIrreversibleBlock: returns the latest irreversible block.
- Call: execute smart contract locally. The tx won't be submitted on chain.
- SendRawTransaction: submit signed transaction. The transaction must be signed before sending.
- GetTransactionReceipt: get transaction receipt info from the transaction hash.

More Nebulas APIs at RPC.

Claim Tokens

Each email can claim tokens every day here.

7.2.2 Tutorials

English

1. Installation (thanks Victor)
2. Sending a Transaction (thanks Victor)
3. Writing Smart Contract in JavaScript (thanks otto)
4. Introducing Smart Contract Storage (thanks Victor)
5. Interacting with Nebulas by RPC API (thanks Victor)

äÿ■æŰĞ

1. çijŰèŕŚåőL'èčĚåŔŁè£ŘèąŇneb
2. åIJÍæŸ§äžŚéŞ¿äÿŁåŔŚéĂĄäžd'æŸŞ
3. ä¡£çŤÍJavaScriptçijŰåĘŹæŹžèČ¡åŘĹçžę
4. æŹžèČ¡åŘĹçžęå■ŸåĆÍåŇžäżŃçż■
5. éĂŽè£ĞRPCæŐěåŔčäÿŐæŸ§äžŚéŞ¿äžd'äžŠ

7.2.3 Contributing

Feel free to join Nebulas Testnet. If you did find something wrong, please submit an issue or submit a pull request to let us know, we will add your name and url to this page as soon as possible.

7.3 Smart Contract

7.3.1 Languages

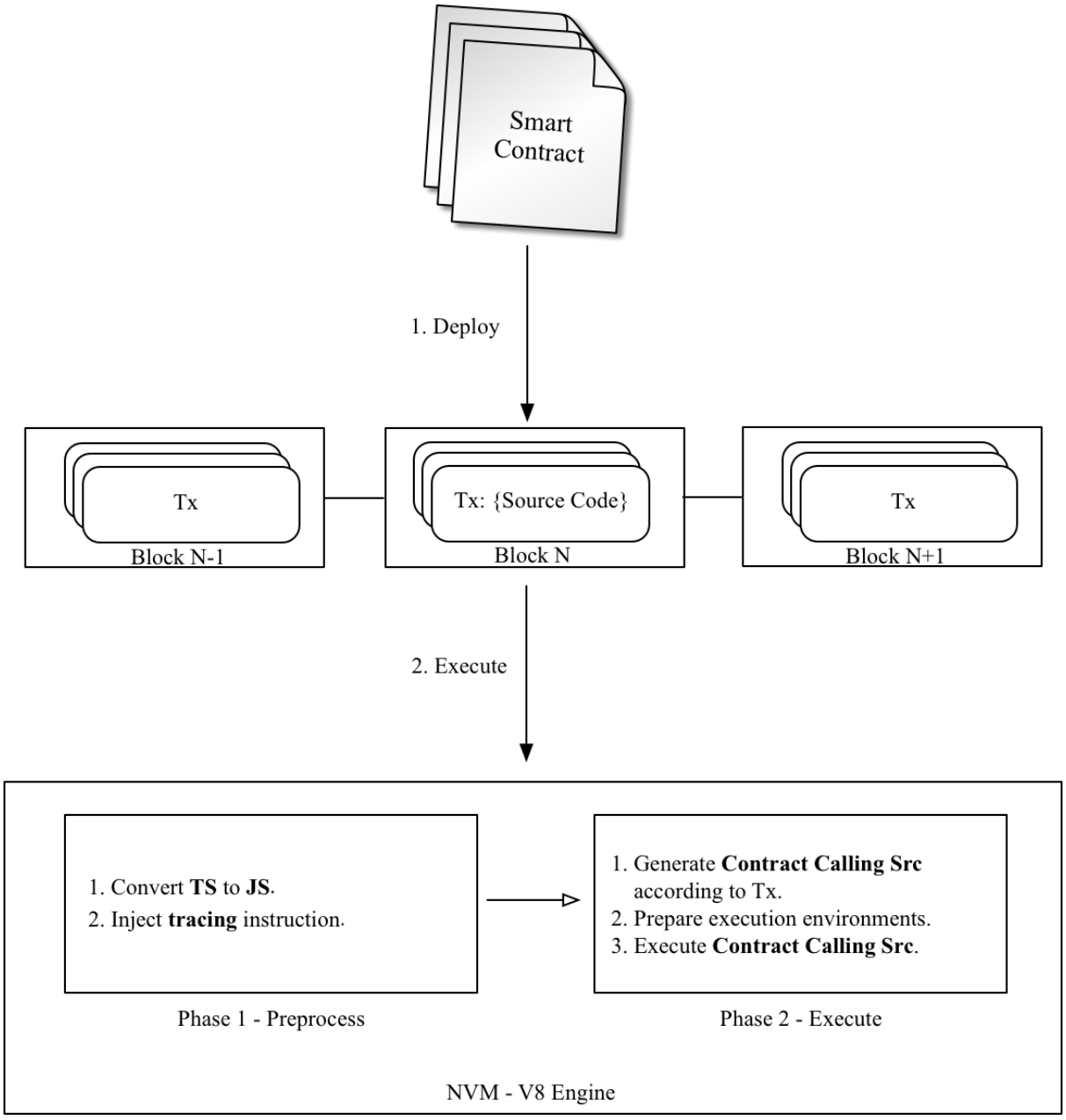
In Nebulas, there are two supported languages for writing smart contracts:

- JavaScript
- TypeScript

They are supported by the integration of Chrome V8, a widely used JavaScript engine developed by The Chromium Project for Google Chrome and Chromium web browsers.

7.3.2 Execution Model

The diagram below is the Execution Model of the Smart Contract:



1. The whole src of the Smart Contract and its arguments are packaged in the Transaction and deployed on Nebulas.
2. The execution of Smart Contract is divided in two phases:
   (a) Preprocess: inject tracing instruction, etc.
   (b) Execute: generate executable src and execute it.

7.3.3 Contracts

Contracts in Nebulas are similar to classes in object-oriented languages. They contain persistent data in state variables and functions that can modify these variables.

Writing Contract

A contract must be a Prototype Object or Class in JavaScript or TypeScript.

A Contract must include an `init` function, it will be executed only once when deploying. Functions whose names start with _ are `private` and can't be executed in a Transaction. The others are all `public` and can be executed in a Transaction.

Since the Contract is executed on Chrome V8, all instance variables are in memory, it's not wise to save all of them to state trie in Nebulas. In Nebulas, we provide `LocalContractStorage` and `GlobalContractStorage` objects to help developers define fields needing to be saved to state trie. And those fields should be defined in `constructor` of the Contract, before other functions.

The following is a sample contract:

```
class Rectangle {
    constructor() {
        // define fields stored to state trie.
        LocalContractStorage.defineProperties(this, {
            height: null,
            width: null,
        });
    }

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

    // calc area function.
    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 + ".");
        }
    }
}
```

Visibility

In JavaScript, there is no function visibility, all functions defined in prototype object are public.

In Nebulas, we define two kinds of visibility `public` and `private`:

- `public` All functions whose name matches the regexp `^[a-zA-Z$][A-Za-z0-9_$]*$` are public, except `init`. Public functions can be called via Transaction.
- `private` All functions whose name starts with _ are private. A private function can only be called by public functions.

7.3.4 Global Objects

console

The `console` module provides a simple debugging console that is similar to the JavaScript console mechanism provided by web browsers.

The global console can be used without calling `require('console')`.

console.info([...args])

- `...args <any>`

  The console.info() function is an alias for `console.log()`.

console.log([...args])

- `...args <any>`

  Print `args` to Nebulas Logger at level `info`.

console.debug([...args])

- `...args <any>`

  Print `args` to Nebulas Logger at level `debug`.

console.warn([...args])

- `...args <any>`

  Print `args` to Nebulas Logger at level `warn`.

console.error([...args])

- `...args <any>`

  Print `args` to Nebulas Logger at level `error`.

LocalContractStorage

The `LocalContractStorage` module provides a state trie based storage capability. It accepts string only key value pairs. And all data is stored to a private state trie associated with the current contract address. Only the contract can access it.

```
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

The `BigNumber` module uses the bignumber.js, a JavaScript library for arbitrary-precision decimal and non-decimal arithmetic operations. The contract can use `BigNumber` directly to handle the value of the transaction and other value transfers.

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

Blockchain

The `Blockchain` module provides an object for contracts to obtain transactions and blocks executed by the current contract. Also, the NAS can be transferred from the contract and the address check is provided.

Blockchain API:

```
// 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`: current block for contract execution
  - `timestamp`: block timestamp
  - `seed`: random seed
  - `height`: block height
- `transaction`: current transaction for contract execution
  - `hash`: transaction hash
  - `from`: sender address of the transaction
  - `to`: recipient address of the transaction
  - `value`: transaction value, a BigNumber object for contract use
  - `nonce`: transaction nonce
  - `timestamp`: transaction timestamp
  - `gasPrice`: transaction gasPrice, a BigNumber object for contract use
  - `gasLimit`: transaction gasLimit, a BigNumber object for contract use
- `transfer(address, value)`: transfer NAS from contract to address
  - params:
    - `address`: nebulas address to receive NAS
    - `value`: transfer value, a BigNumber object
  - return:

        * `0`: transfer success
        * `1`: transfer failed
- `verifyAddress(address)`: verify address
    - params:
        * `address`: address need to check
    - return:
        * `1`: address is valid
        * `0`: address is invalid

Example to use:

```
'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

The `Event` module records execution events in the contract. The recorded events are stored in the event trie on the chain, which can be fetched by `FetchEvents` method in block with the execution transaction hash. All contract event topics have a `chain.contract.` prefix before the topic they set in contract.

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

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

You can see the example in `SampleContract` above.

Math.random

- `Math.random()` returns a floating-point, pseudo-random number in the range from 0 inclusive, up to, but not including 1. The typical usage is:

```
"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)` if needed, you can use this method to reset the random seed. The argument `myseed` must be a **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:

- Unsupported methodsïijŽ`toDateString()`, `toTimeString()`, `getTimezoneOffset()`,`toLocaleXXX()`.
- `new Date()`/`Date.now()` returns the timestamp of current block in milliseconds.
- `getXXX` returns the result of `getUTCXXX`.

accept

this method aims to make it possible to send a binary transfer to a contract account. As `to` is a smart contact address, which has declared the function `accept()` and it excuted correctly, the transfer will succeed. If the Tx is a non-binary Tx, it will be treated as a normal function.

```
"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;
```

7.4 NRC20

7.4.1 Abstract

The following standard allows for the implementation of a standard API for tokens within smart contracts. This standard provides basic functionality to transfer tokens, as well as allows tokens to be approved so they can be spent by another on-chain third party.

7.4.2 Motivation

A standard interface allows that a new token can be created by any application easily : from wallets to decentralized exchanges.

7.4.3 Methods

name

Returns the name of the token - e.g. `"MyToken"`.

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

symbol

Returns the symbol of the token. E.g. “TK”.

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

decimals

Returns the number of decimals the token uses - e.g. `8`, means to divide the token amount by `100000000` to get its user representation.

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

totalSupply

Returns the total token supply.

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

balanceOf

Returns the account balance of a address.

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

transfer

Transfers `value` amount of tokens to `address`, and MUST fire the `Transfer` event. The function SHOULD `throw` if the `from` account balance does not have enough tokens to spend.

Note Transfers of 0 values MUST be treated as normal transfers and fire the `Transfer` event.

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

transferFrom

Transfers `value` amount of tokens from address `from` to address `to`, and MUST fire the `Transfer` event.

The `transferFrom` method is used for a withdraw workflow, allowing contracts to transfer tokens on your behalf. This can be used for example to allow a contract to transfer tokens on your behalf and/or to charge fees in sub-currencies. The function SHOULD `throw` unless the `from` account has deliberately authorized the sender of the message via some mechanism.

Note Transfers of 0 values MUST be treated as normal transfers and fire the `Transfer` event.

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

approve

Allows `spender` to withdraw from your account multiple times, up the `currentValue` to the `value` amount. If this function is called again it overwrites the current allowance with `value`.

NOTE: To prevent attack vectors, the user needs to give a previous approve value, and the default value that is not approve is 0.

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

allowance

Returns the amount which `spender` is still allowed to withdraw from `owner`.

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

Events

transferEvent

MUST trigger when tokens are transferred, including zero value transfers.

A token contract which creates new tokens SHOULD trigger a Transfer event with the `from` address set to `totalSupply` when tokens are created.

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

approveEvent

MUST trigger on any call to `approve(spender, currentValue, value)`.

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

7.4.4 Implementation

Example implementations are available at

- 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;
```

7.5 Tools

All the developing tools: official dev tools and tools from the community. Welcome to build the Nebulas ecosystem together. You can recommend more tools and edit this page on Github directly.

- **Cross-platform Nebulas smart contract IDE**

Full functions: web

Local NVM: Mac OS, Windows, Linux

- **nebPay**

Nebulas payment JavaScript API. Users can use it in browser on both PC and mobile. Users can do NAS payment through Chrome extension and iOS/Android wallet with it.

- **Development Environment for Nebulas**

7.5.1 JavaScript development tools

- **VS Code**
- **sublime**

7.5.2 DApp development framework

- **Nasa.js** The acclaimed Nebulas DApp client development framework, lightweight and easy to use.
- **Nebulas DApp Local Development Debugging Tool**

7.5.3 Contract development tools

- **Smart contract integrated development environment**
- **Nebulas smart contract ide**

7.5.4 Contract deployment tool

- **Web-wallet**
- **WebExtensionWallet**

7.5.5 Nebpay

- **JavaScript SDK**
- **iOS SDK**
- **Android SDK**

7.5.6 Nebulas API

- **Go**
- **Python**

- **PHP**
- **ruby**
- **NET**
- **unity3d**
- **swift**

7.5.7 Static scan tool

- **Nebulas Smart Contract Code Checker**
- **NebulasÂăSmart Contract Lint Tool**
- **Nebulas javascript/typescript smart contract static check tool**

7.5.8 Command line tool

- **A CLI Tool forÂăNebulas**

7.5.9 test tools

- **NebTest will automate unit testing ofÂănebulasÂăsmart contracts**

7.5.10 other

NebulasDB is a nebulas-based, decentralized, non-relational database, and provides JS-SDK, client

- **The console is easy to develop for data operations**
- **Nebulas-Utils is an utiliy package for Nebulas Chain Development**
- **Base onÂăNebulasÂăJs API, putÂănebulas.js and nebpay.js on one package**

7.6 DApps Design Guide

TBA

You can download PDF here.

7.7 Learning Resources

All learning resources. Videos and documents. Welcome to recommend more resources from the community, and you can edit this page on Github directly. Help others and learn things together.

7.7.1 Official Nebulas Documents

- Nebulas Mauve Paper: Developer Incentive Protocol (English)
- Nebulas Mauve Paper: Developer Incentive Protocol (Chinese)
- About Nebulas Mauve Paper: Developer Incentive Protocol
- Nebulas Rank Yellow Paper (English)
- Nebulas Rank Yellow Paper (Chinese)
- Nebulas Rank Yellow Paper (Korean)
- Nebulas Rank Yellow Paper (Portuguese)
- Official Interpretation of âĂIJNebulas Rank Yellow PaperâĂİ
- Technical Whitepaper (English)
- Technical Whitepaper (Chinese)
- Non-technical Whitepaper (English)
- Non-technical Whitepaper (Chinese)

7.7.2 Dive into Nebulas

- Dive into Nebulas 1 âĂŤ An Introduction
- Dive into Nebulas 2 âĂŤ A Quick Start
- Dive into Nebulas 3âĂŁâĂŤâĂŁManaging Accounts
- Dive into Nebulas 4âĂŤ Transactions

7.7.3 How to build a DApp on Nebulas

- How to build a DApp on Nebulas (Part 1)
- How to build a DApp on Nebulas (Part 2)
- How to build a DApp on Nebulas (Part 3)
- Details on the Smart Contract Ranking Algorithm Part 1
- Details on the Smart Contract Ranking Algorithm Part 2

- New Nebulas Smart Contract feature
- Claim Nebulas Testnet Token Step by Step
- Why Choose Nebulas at a Hackathon?
- How to architect a DApp using Nuxt.js and Nebulas by Honey Thakuria
- Nebulas: JavaScript Meets Smart Contracts âĂŤâĂŤ An Intro to Nebulas for Ethereum Smart Contract Developers by Michal Zalecki

7.7.4 How to use Nebulas Wallet

1. Creating A NAS Wallet Nebulas Wallet
2. Sending NAS from your Wallet Nebulas Wallet
3. Signing a Transaction Offline Nebulas Wallet
4. View Wallet Information Nebulas Wallet
5. Check TX Status Nebulas Wallet
6. Deploy a Smart Contract Nebulas Wallet
7. Call a Smart Contract on Nebulas Nebulas Wallet

- How to use NebPay in your Dapp

7.7.5 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

7.8 RPC Overview

Remote Procedure Calls (RPCs) provide a useful abstraction for building distributed applications and services.

Nebulas provides both gRPC and RESTful API for users to interact with Nebulas.

grpc provides a concrete implementation of the gRPC protocol, layered over HTTP/2. These libraries enable communication between clients and servers using any combination of the supported languages.

grpc-gateway is a plugin of protoc. It reads gRPC service definition, and generates a reverse-proxy server which translates a RESTful JSON API into gRPC. we use it to map gRPC to HTTP.

7.8.1 Endpoint

Default endpoints:

API	URL	Protocol
gRPC	http://localhost:8684	Protobuf
RESTful	http://localhost:8685	HTTP

gRPC API

We can run the gRPC example testing client code:

```
go run main.go
```

The testing client gets account state from sender address, makes a transaction from sender to receiver, and also checks the account state of receiver address.

We can see client log output like:

```
GetAccountState n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5 nonce 4 value␣
↪314283103999999999992
SendTransaction n1QZMXSZtW7BUerroSms4axNfyBGyFGkrh5 ->␣
↪n1Zn6iyyQRhqthmCfqGBzWfip1Wx8wEvtrJ value 2 txhash:
↪"2c2f5404a2e2edb651dff44a2d114a198c00614b20801e58d5b00899c8f512ae"
GetAccountState n1Zn6iyyQRhqthmCfqGBzWfip1Wx8wEvtrJ nonce 0 value 10
```

HTTP

Now we also provided HTTP to access the RPC API. The file that ends with **gw.go** is the mapping file. Now we can access the rpc API directly from our browser, you can update the **rpc_listen** and **http_listen** in **conf/default/config.conf** to change RPC/HTTP port.

Example:

```
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..."
}
```

7.8.2 RPC methods

- GetNebState
- GetAccountState
- LatestIrreversibleBlock
- Call
- SendRawTransaction
- GetBlockByHash
- GetBlockByHeight
- GetTransactionReceipt
- GetTransactionByContract
- GetGasPrice
- EstimateGas
- GetEventsByHash
- Subscribe
- GetDynasty

7.8.3 RPC API Reference

GetNebState

Return the state of the neb.

Protocol	Method	API
gRpc		GetNebState
HTTP	GET	/v1/user/nebstate

Parameters

none

Returns

`chain_id` Block chain id

`tail` Current neb tail hash

`lib` Current neb lib hash

`height` Current neb tail block height

`protocol_version` The current neb protocol version.

`synchronized` The peer sync status.

`version` neb version.

HTTP Example

```
// 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 the state of the account. Balance and nonce of the given address will be returned.

Protocol	Method	API
gRpc		GetAccountState
HTTP	POST	/v1/user/accountstate

Parameters

`address` Hex string of the account addresss.

`height` block account state with height. If not specified, use 0 as tail height.

Returns

`balance` Current balance in unit of 1/(10^18) nas.

`nonce` Current transaction count.

`type` The type of address, 87 stands for normal address and 88 stands for contract address

HTTP Example

```
// Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/user/accountstate -d '{"address":
↪"n1Z6SbjLuAEXfhX1UJvXT6BB5osWYxVg3F3"}'

// Result
{
    result {
        "balance":"94899999989800000000"
        "nonce":51
        "type":87
    }
}
```

LatestIrreversibleBlock

Return the latest irreversible block.

Protocol	Method	API
gRpc		LatestIrreversibleBlock
HTTP	GET	/v1/user/lib

Parameters

none

Returns

`hash` Hex string of block hash.

`parent_hash` Hex string of block parent hash.

`height` block height.

`nonce` block nonce.

`coinbase` Hex string of coinbase address.

`timestamp` block timestamp.

`chain_id` block chain id.

`state_root` Hex string of state root.

`txs_root` Hex string of txs root.

`events_root` Hex string of event root.

`consensus_root`

- `Timestamp` time of consensus state
- `Proposer` proposer of current consensus state
- `DynastyRoot` Hex string of dynasty root

`miner` the miner of this block

`is_finality` block is finality

`transactions` block transactions slice.

- `transaction` GetTransactionReceipt response info.

HTTP Example

```
// 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

Call a smart contract function. The smart contract must have been submited. Method calls are run only on the current node, not broadcast.

Protocol	Method	API
gRpc		Call
HTTP	POST	/v1/user/call

Parameters

The parameters of the `call` method is the same as the SendTransaction parameters. Special attention:

`to` Hex string of the receiver account addresss. **The value of**`to`** is a contract address.**

`contract` transaction contract object for call smart contract.

- Sub properties(`source`** and **‘‘sourceType‘‘** are not need**):
  - `function` the contract call function for call contarct function.
  - `args` the params of contract. The args content is JSON string of parameters array.

Returns

`result` result of smart contract method call

`execute_err` execute error

`estimate_gas` estimate gas used

HTTP Example

```
// Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/user/call -d '{"from":
↪"n1Z6SbjLuAEXfhX1UJvXT6BB5osWYxVg3F3","to":
↪"n1mLzWCzyR1fUELEugfCZoNAW3dt8QpHtJw","value":"0","nonce":3,
↪"gasPrice":"1000000","gasLimit":"2000000","contract":{"function":
↪"transferValue","args":"[500]"}}'
```

```
// Result
{
   "result": "0",
   "execute_err": "insufficient balance",
   estimate_gas: "22208"
}
```

SendRawTransaction

Submit the signed transaction. The transaction signed value should be return by SignTransactionWithPassphrase.

Protocol	Method	API
gRpc		SendRawTransaction
HTTP	POST	/v1/user/rawtransaction

Parameters

`data` Signed data of transaction

Returns

`txhash` Hex string of transaction hash.

`contract_address` returns only for deploy contract transaction.

HTTP Example

```
// 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"
    }
}
```

Deploy Contract Example

```
// 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

Get block header info by the block hash.

Protocol	Method	API
gRpc		GetBlockByHash
HTTP	POST	/v1/user/getBlockByHash

Parameters

`hash` Hex string of transaction hash.

`full_fill_transaction` If true it returns the full transaction objects, if false only the hashes of the transactions.

Returns

See LatestIrreversibleBlock response.

HTTP Example

```
// Request
curl -i -H 'Content-Type: application/json' -X POST http://
↪localhost:8685/v1/user/getBlockByHash -d '{"hash":
↪"00000658397a90df6459b8e7e63ad3f4ce8f0a40b8803ff2f29c611b2e0190b8
↪", "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

Get block header info by the block height.

Protocol	Method	API
gRpc		GetBlockByHeight
HTTP	POST	/v1/user/getBlockByHeight

Parameters

`height` Height of transaction hash.

`full_fill_transaction` If true it returns the full transaction objects, if false only the hashes of the transactions.

Returns

See LatestIrreversibleBlock response.

HTTP Example

```
// 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

Get transactionReceipt info by tansaction hash. If the transaction not submit or only submit and not packaged on chain, it will reurn not found error.

Protocol	Method	API
gRpc		GetTransactionReceipt
HTTP	POST	/v1/user/getTransactionReceipt

Parameters

`hash` Hex string of transaction hash.

Returns

`hash` Hex string of tx hash.

`chainId` Transaction chain id.

`from` Hex string of the sender account addresss.

`to` Hex string of the receiver account addresss.

`value` Value of transaction.

`nonce` Transaction nonce.

`timestamp` Transaction timestamp.

`type` Transaction type.

`data` Transaction data, return the payload data.

`gas_price` Transaction gas price.

`gas_limit` Transaction gas limit.

`contract_address` Transaction contract address.

`status` Transaction status, 0 failed, 1 success, 2 pending.

`gas_used` transaction gas used

HTTP Example

```
// 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"
    }
}
```

GetTransactionByContract

Get transactionReceipt info by contract address. If contract not exists or packaged on chain, a not found error will be returned.

Protocol	Method	API
gRpc		GetTransactionByContract
HTTP	POST	/v1/user/getTransactionByContract

Parameters

`address` Hex string of contract account address.

Returns

The result is the same as that of GetTransactionReceipt

HTTP Example

```
// 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 the subscribed events of transaction & block. The request is a keep-alive connection.

Protocol	Method	API
gRpc		Subscribe
HTTP	POST	/v1/user/subscribe

Parameters

`topics` repeated event topic name, string array.

The topic name list:

- `chain.pendingTransaction` The topic of pending a transaction in transaction_pool.
- `chain.latestIrreversibleBlock` The topic of updating latest irreversible block.
- `chain.transactionResult` The topic of executing & submitting tx.
- `chain.newTailBlock` The topic of setting new tail block.
- `chain.revertBlock` The topic of reverting block.

Returns

`topic` subscribed event topic name.

`data` subscribed event data.

HTTP Example

```
// 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\": \"1000000\",
                 \"gaslimit\":\"20000000\",
                 \"type\":\"deploy\"}"
    }
    "result":{
        "topic":"chain.pendingTransaction",
        "data": "..."
    }
    ...
}
```

GetGasPrice

Return current gasPrice.

Protocol	Method	API
gRpc		GetGasPrice
HTTP	GET	/v1/user/getGasPrice

Parameters

none

Returns

`gas_price` gas price. The unit is 10^-18 NAS.

HTTP Example

```
// Request
curl -i -H 'Content-Type: application/json' -X GET http://
↪localhost:8685/v1/user/getGasPrice

// Result
{
    "result":{
        "gas_price":"1000000"
    }
}
```

EstimateGas

Return the estimate gas of transaction.

Protocol	Method	API
gRpc		EstimateGas
HTTP	POST	/v1/user/estimateGas

Parameters

The parameters of the `EstimateGas` method is the same as the SendTransactionparameters.

Returns

`gas` the estimate gas.

`err` error message of the transaction executing

HTTP Example

```
// 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":"1000000","gasLimit":
↪"2000000"}'

// Result
{
    "gas":"20000",
    "err":""
}
```

GetEventsByHash

Return the events list of transaction.

Protocol	Method	API
gRpc		GetEventsByHash
HTTP	POST	/v1/user/getEventsByHash

Parameters

`hash` Hex string of transaction hash.

Returns

`events` the events list.

- `topic` event topic;
- `data` event data.

HTTP Example

```
// 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 get dpos dynasty.

Protocol	Method	API
gRpc		GetDynasty
HTTP	POST	/v1/user/dynasty

Parameters

`height` block height

Returns

`miners` repeated string of miner address.

HTTP Example

```
// 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"
            ]
        }
    }
}
```

Management RPC

Beside the NEB API RPC interface nebulas provides additional management APIs. Neb console supports both API and management interfaces. Management RPC uses the same gRPC and HTTP port, which also binds NEB API RPC interfaces.

Nebulas provide both gRPC and RESTful management APIs for users to interact with Nebulas. Our admin proto file defines all admin APIs. **We recommend using the console access admin interfaces, or restricting the admin RPC to local access.**

Default management RPC Endpoint:

| API | URL | Protocol | | — | :—: | :—: | | gRPC | http://localhost:8684 | Protobuf | | RESTful | http://localhost:8685 | HTTP |

Management RPC methods

- NodeInfo
- Accounts
- NewAccount
- UnLockAccount
- LockAccount
- SignTransactionWithPassphrase
- SendTransactionWithPassphrase
- SendTransaction
- SignHash
- StartPprof
- GetConfig

Management RPC API Reference

NodeInfo

Return the p2p node info.

| Protocol | Method | API | | — | — | — | | gRpc | | NodeInfo | | HTTP | GET | /v1/user/nodeinfo |

Parameters

none

Returns

`id` the node ID.

`chain_id` the block chainID.

`coninbase` coinbase

`peer_count` Number of peers currenly connected.

`synchronized` the node synchronized status.

`bucket_size` the node route table bucket size.

`protocol_version` the network protocol version.

`RouteTable*[] route_table` the network routeTable

```
message RouteTable {
    string id = 1;
    repeated string address = 2;
}
```

HTTP Example

```
# 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 account list.

| Protocol | Method | API | | — | — | — | | gRpc | | Accounts | | HTTP | GET | /v1/admin/accounts |

Parameters

none

Returns

`addresses` account list

HTTP Example

```
# 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 create a new account with passphrase.

| Protocol | Method | API | | — | — | — | | gRpc | | NewAccount | | HTTP | POST | /v1/admin/account/new |

Parameters

`passphrase` New account passphrase.

Returns

`address` New Account address.

HTTP Example

```
# 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 unlock account with passphrase. After the default unlock time, the account will be locked.

| Protocol | Method | API | | — | — | — | | gRpc | | UnLockAccount | | HTTP | POST | /v1/admin/account/unlock |

Parameters

`address` UnLock account address.

`passphrase` UnLock account passphrase.

`duration` Unlock accoutn duration.

Returns

`result` UnLock account result, unit is ns.

HTTP Example

```
# 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 lock account.

| Protocol | Method | API | | — | — | — | | gRpc | | LockAccount | | HTTP | POST | /v1/admin/account/lock |

Parameters

`address` Lock account address.

Returns

`result` Lock account result.

HTTP Example

```
# 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 sign transaction. The transaction's from addrees must be unlocked before sign call.

| Protocol | Method | API | | — | — | — | | gRpc | | SignTransactionWithPassphrase | | HTTP | POST | /v1/admin/sign |

Parameters

`transaction` this is the same as the SendTransaction parameters.

`passphrase` from account passphrase

Returns

`data` Signed transaction data.

sign normal transaction Example

```
# 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":"1000000","gasLimit":
↪"2000000"}, "passphrase":"passphrase"}'

# Result
{
    "result":{
        "data":
↪"CiBOW15yoZ+XqQbMNr4bQdJCXrBTehJKukwjcfW5eASgtBIaGVduKnw+6lM3HBXhJEzzuvv3yNdYA
↪BwhwhqUkp/
↪gEJtE4kndoc7NdSgqD26IQqa0Hjbtg1JaszAvHZiW+XH7C+Ky9XTKRJKuTOc446646d/
↪Sbz/nxQE="
    }
}
```

SendTransactionWithPassphrase

SendTransactionWithPassphrase send transaction with passphrase.

| Protocol | Method | API | | — | — | — | | gRpc | | SendTransactionWithPassphrase | | HTTP | POST | /v1/admin/transactionWithPassphrase |

Parameters

`transaction` transaction parameters, which is the same as the SendTransaction parameters.

`passphrase` From address passphrase.

Returns

`txhash` transaction hash.

`contract_address` returns only for deploy contract transaction.

Example

```
# 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":"1000000","gasLimit":
↪"2000000"},"passphrase":"passphrase"}'

# Result
{
    "result":{
        "hash":
↪"143eac221da8079f017bd6fd6b6a08ea0623114c93c638b94334d16aae109666
↪",
        "contract_address":""
    }
}
```

SendTransaction

Send the transaction. Parameters `from`, `to`, `value`, `nonce`, `gasPrice` and `gasLimit` are required. If the transaction is to send contract, you must specify the `contract`.

| Protocol | Method | API | | — | — | — | | gRpc | | SendTransaction | | HTTP | POST | /v1/user/transaction |

Parameters

`from` Hex string of the sender account addresss.

`to` Hex string of the receiver account addresss.

`value` Amount of value sending with this transaction.

`nonce` Transaction nonce.

`gas_price` gasPrice sending with this transaction.

`gas_limit` gasLimit sending with this transaction.

`type` transaction payload type. If the type is specified, the transaction type is determined and the corresponding parameter needs to be passed in, otherwise the transaction type is determined according to the contract and binary data. [optional]

- type enum:
  - `binary`: normal transaction with binary

  - `deploy`: deploy smart contract
  - `call`: call smart contract function

`contract` transaction contract object for deploy/call smart contract. [optional]

- Sub properties:
  - `source` contract source code for deploy contract.
  - `sourceType` contract source type for deploy contract. Currently support `js` and `ts`
    * `js` the contract source write with javascript.
    * `ts` the contract source write with typescript.
  - `function` the contract call function for call contarct function.
  - `args` the params of contract. The args content is JSON string of parameters array.

`binary` any binary data with a length limit = 64bytes. [optional]

Notice:

- `from = to` when deploy a contract, the `to` address must be equal to `from` address.
- `nonce` the value is **plus one**(+1) on the nonce value of the current from address. Current nonce can get from GetAccountState.
- `gasPrice` and `gasLimit` need for every transaction. We recommend taking them use GetGasPrice and EstimateGas.
- `contract` parameter only need for smart contract deploy and call. When a smart contract is deployed, the `source` and `sourceType` must be specified, the `args` is optional and passed in when the initialization function takes a parameter. The `function` field is used to call the contract method.

Returns

`txhash` transaction hash.

`contract_address` returns only for deploying contract transaction.

Normal Transaction Example

```
# 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":"1000000","gasLimit
↪":"2000000"}'

# Result
{
    "result":{
      "txhash":
↪"fb5204e106168549465ea38c040df0eacaa7cbd461454621867eb5abba92b4a5
↪",
```

```
      "contract_address":""
    }
}
```

Deploy Smart Contract Example

```
# 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":"1000000","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)}
↪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;","sourceType":"js",
↪"args":""}}'

# Result
{
    "result":{
        "txhash":
↪"3a69e23903a74a3a56dfc2bfbae1ed51f69debd487e2a8dea58ae9506f572f73
↪",
        "contract_address":"n21Y7arNbUfLGL59xgnA4ouinNxyvz773NW"
    }
}
```

SignHash

SignHash sign the hash of a message.

| Protocol | Method | API | | — | — | — | | gRpc | | SignHash | | HTTP | POST | /v1/admin/sign/hash |

Parameters

`address` Sign address

`hash` A sha3256 hash of the message

`alg` Sign algorithm

Returns

`data` Signed transaction data.

sign normal transaction Example

```
# 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

| Protocol | Method | API | | — | — | — | | gRpc | | Pprof | | HTTP | POST | /v1/admin/pprof |

Parameters

`listen` the address to listen

Returns

`result` start pprof result

Example

```
# 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 the config current neb is using

| Protocol | Method | API | | — | — | — | | gRpc | | GetConfig | | HTTP | GET | /v1/admin/getConfig |

Parameters

none

Returns

`config` neb config

Example

```
# 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"
            }
        }
    }
}
```

7.9 REPL console

Nebulas provide an interactive javascript console, which can invoke all API and management RPC methods. The console is connected to the local node by default without specifying host.

7.9.1 start console

Start console using the command:

```
./neb console
```

In the case of not specifying the configuration file, the terminal's startup defaults to the configuration of `conf/default/config.conf`. If the local configuration file is not available or you want to specify the configuration file, the terminal starts like this:

```
./neb -c <config file> console
```

console interaction

The console can use the `admin.setHost` interface to specify the nodes that are connected. When the console is started or the host is not specified, the terminal is interacting with the local node. **Therefore, you need to start a local node before starting the console.**

```
> admin.setHost("https://testnet.nebulas.io")
```

Tips: The Testnet only starts the RPC interface of the API, so only the api scheme is available.

7.9.2 console usage

We have API and admin two schemes to access the console cmds. Users can quickly enter instructions using the `TAB` key.

```
> 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
```

Some management methods may require passphrase. The user can pass in the password when the interface is called, or the console prompts the user for input when the password is not entered. **We recommend using a console prompt to enter your password because it is not visible.**

Enter the password directly:

```
> admin.unlockAccount("n1UWZa8yuvRgePRPgp8a2jX4J9UwGXfHp6i",
↪"passphrase")
{
```

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

Use terminal prompt:

```
> admin.unlockAccount("n1UWZa8yuvRgePRPgp8a2jX4J9UwGXfHp6i")
Unlock account n1UWZa8yuvRgePRPgp8a2jX4J9UwGXfHp6i
Passphrase:
{
    "result": {
        "result": true
    }
}
```

The interfaces with passphrase prompt:

```
admin.newAccount
admin.unlockAccount
admin.signHash
admin.signTransactionWithPassphrase
admin.sendTransactionWithPassphrase
```

The command parameters of the command line are consistent with the parameters of the RPC interface. NEB RPC and NEB RPC_Admin.

7.9.3 console exit

The console can exit with the `ctrl-C` or `exit` command.

CHAPTER 8

Community

8.1 Dynamics

- (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

8.2 Events

Since June 2017, the Nebulas meetups and hackathons have been held in 17 cities, 8 countries around the world. We have visited the University of California, Berkeley, the New York University, Columbia University, Harvard University, the Singapore University of Social Sciences, Tsinghua University, Tongji University, and many others.

Events History >

You are welcome to organize local meetups and participate in the history of Nebulas!

8.3 Announcement

- (12/05/2018) - Announcement: Nebulas Testnet Operation Maintenance
- (11/27/2018) - Nebulas Bug Bounty Improvement
- (10/17/2018) - ATP Smartdrop Applying Process Begins
- (10/15/2018) - Nebulas Mainnet Snapshot of ATP Smartdrop Ended
- (10/10/2018) - Announcement on Token Swap Ends via NAS nano
- (10/08/2018) - Claim Instructions of ATP Smartdrop
- (09/25/2018) - Upgrading NAS nano to Version 2.2.0
- (09/25/2018) - Announcing the NAS Token Swap via NAS nano v2.2.0
- (09/21/2018) - Nebulas Nova: âĂIJDiscover Value in an Organized Blockchain WorldâĂİ
- (09/13/2018) - Announcing the Nebulas Technical Committee
- (09/01/2018) - Announcing Unreleased NAS Locking Addresses
- (08/30/2018) - Increasing Bug Bounty Rewards for Inter-contract Call Functions Testing
- (08/22/2018) - Nebulas Testnet Inter-contract Call Function Public Beta Bounty
- (08/21/2018) - Nebulas Mainnet Security Upgrade Notice
- (08/08/2018) - Announcing the Adjustment of Reserved NAS Distribution to the Nebulas Team
- (06/29/2018) - Nebulas Mainnet Token Swap Announcement
- (06/28/2018) - Nebulas Rank Yellow PaperâĂŁâĂŤâĂŁBlockchain 3.0 Primer
- (06/20/2018) - Nebulas Mainnet Upgrade
- (05/11/2018) - Supplementary explanation for NAS mainnet coinswap
- (04/30/2018) - Announcement on NAS mainnet coin swap starts in exchanges
- (04/25/2018) - Statement on Asset Security from the Nebulas Tech Team
- (04/13/2018) - Important Announcement on the Nebulas Lock Up Bonus Program
- (03/29/2018) - Nebulas Foundation Notice
- (03/27/2018) - Nebulas Testnet Upgrade Announcement
- (12/13/2017) - Disclaimer Regarding Private Pools for NAS Pre-Sale
- (12/13/2017) - New Pricing Rules for NAS Pre-Sale Early Bird Participants
- (11/24/2017) - Initiation of âĂIJNAS Token Bonus ProgramâĂİ

8.4 Weekly Report

You can take a quick look for all the Nebulas weekly reports here.

- 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

8.5 Ask Me Anything

8.5.1 Nebulas Reddit AMA

- Reddit Questions and Answers - Reddit Weekly Question Recap! (10.29–11.4)
- Reddit Questions and Answers - Reddit Weekly Discussion Recap!ïijĹ10.21–10.26ïijL’
- Nebulas Reddit AMA Recap - With Nebulas Founder Hitters Xu and Co-founder Aero Wang
- Reddit Questions and Answers - Nebulas Weekly AMA & Constructive SuggestionsâĂŁâĂŤâĂŁAugust 6 to August 19
- Reddit Questions and Answers - Nebulas Weekly AMA & Constructive SuggestionsâĂŁâĂŤâĂŁJuly 30 to August 6 2018
- Reddit Questions and Answers - Nebulas Weekly AMA & Constructive SuggestionsâĂŁâĂŤâĂŁJuly 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

8.5.2 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?

8.6 Nebulas Interviews

8.6.1 Interviews with Nebulas Team

- Interview with Nebulas Team - Nebulers' Thoughts on the Future of Blockchain

8.6.2 Interviews with Members of Nebulas Research Institute

- Interview with the leader of Nebulas Research Institute Dr. Xuepeng Fan - Take the Lead to Set Up Nebulas Research Institute
- Interview with Nebulas Mainnet Development Lead Dr. Congming Chen - Let Nebulas Fly Higher and Farther!
- Interview with Nebulas Senior Researcher Dr. Zaiyang Tang - My Heart Belongs to Nebulas, I Hope We Shine Together
- Interview with Nebulas Technical Director Dr. Joel - Exclusive Interview to Nebulas Technical Director Dr. Joel
- Interview with the Senior Researcher of Nebulas Research Institute Dr. Yulong Zeng - My First Job at Nebulas
- Interview with Nebulas Research Institute Intern Dr. Dai - Life Is A Challenge

8.6.3 Interviews with Members of the Nebulas Technical Committee

- Interview with the CEO and Founder of Nebulas Hitters Xu - Seeing Through The Blockchain Bubble: Sitting Down For An Interview With Nebulas.io Founder Hitters Xu

8.6.4 Interviews with Members of the Community

- Interview with Pluto and Xuxue (Nebulas Incentive Program Week 1 Champions) - Nebulas Incentive ProgramâĂŁâĂŤâĂŁInterview with the Champion of Week 1
- Interview with Jason Mansfield (Multi-time winner of Nebulas Incentive Program Season 1) - Interview with a Nebulas DApp Developer: Jason Mansfield
- Interview with Honey Thakuria (Accenture Hackathon Winner using Nebulas) - DApp Development and Architecture DesignâĂŁâĂŤâĂŁInterview with Honey Thakuria

CHAPTER 9

Ecosystem

9.1 NAS nano

NAS nano is the official wallet. Please download it here

9.2 Nebulas Web Wallet Tutorial

- Part 7 - Call a Smart Contract on Nebulas Wallet
- Part 6 - Deploy a Smart Contract
- Part 5 - Check TX Status
- Part 4 - View Wallet Information
- Part 3 - Signing a Transaction Offline
- Part 2 - Sending NAS from your Wallet
- Part 1 - Creating A NAS Wallet

9.3 NAS nano Help Document is Coming Soon!

9.4 Ecosystem DApps List

You can find recommended DApps and the monthly/weekly champions of the First Season of the Nebulas Incentive Program here: Summary

You are welcome to recommend more DApps!

The Nebulas DApps Store by the community.

CHAPTER 10

Useful Links

Tutorials made by the nebulas community:

How to architect a DApp using Nuxt.js and Nebulas

A walkthrough to your first deployed dApp

Nebulearn

Nebulas Smart Contract Tutorial

Nebulas Dapp Tutorial: WebExtensionWallet and NebPay.js

Nebulas Dapp tutorial

NNew Nebulas Smart Contract feature

Creating A NAS Wallet

CHAPTER 11

Frequently Asked Questions

TBA

Please visit here and you can also get more information on Reddit

CHAPTER 12

Licensing

12.1 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.

12.2 Contributor License Agreement

TBD.

12.2.1 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
}
```

12.2.2 Contributors

Coming Soon.

CHAPTER 13

Indices and tables

- genindex
- modindex
- search