ZeusNetChain

API Specification

Build Record

Build No	Editor	Approver	Modified Date	Comments
0.0.1	Ailen		2017/7/17	Created
0.0.2	Ailen		2017/11/2	Modified
0.0.3	Jack		2018/1/20	Modified

1 Introduction

1.1 Purpose

This document describes the interfaces that are used for interacting with ZeusNetChain. It serves as a standard and a guideline for the interaction between the user's own system and the ZeusNetChain to guide the developers in designing, developing, and testing interfaces.

1.2 ZeusNetChain Overview

ZeusNetChain is a completely independent developed and highly efficient implementation of blockchain. It is the core engine of ZeusNet and its soul. ZeusNet is a highly efficient computing resource management and scheduling system. It does quick matching of global computing resource transactions. It dispatches the Đapp running environment to designated computing resource hardware through its accurate scheduling and fast distribution. And provides fair and reliable work load statistics & settlement for computing resource leasing transactions all over the world. All these rely on a powerful and efficient implementation of blockchain technology. Currently available blockchain 3.0 products on the market cannot meet the demand. ZeusNetChain are the technical implementation of Blockchain 4.0. Its performance, scalability, security and stability features are significantly better than those of Blockchain 3.0 technology.

ZeusNetChain is a blockchain platform meets the high-performance transaction and multi-purpose computing capabilities under the Token value system. ZeusNetChain has a high-performance main chain for trading transactions, plus a number of ecological chain to achieve the decentralization of different computing power resources.

ZeusNetChain’s core strengths include:

1) 1+9 Multi-Chain Technology: a main transaction chain, to achieve a variety of services and the blockchain registration, registration, query. Computing needs match are done by the other nine chains. It utilizes the stochastic correlation analysis of the Markov chain, to achieve quick block creation and confirmation of different transactions to the timing.

2) DDN: Docker Deliver Network. The key question for computing power trading market is to distribute the Đapps running environment to the designated computing resource after the lease of assets has been agreed. Then start official start the computing resources consumption, and seeking a balance between the fairness and efficiency. ZeusNetChain utilizes features like service delivery path optimization, cloud computing, and Docker container delivery in its architecture design process, to achieve a balance of fairness and efficiency.

3) Token Issue Mechanism: ZeusNet uses ZNC as the main currency, can “fork” to other tokens, and other tokens can anchor ZNC to exchange with other digital currencies such as BTC / ETH. For instance, a CDN scenario can issue CDN token; a GPU computing power scenario can issue a GToken; different games may issue different game tokens (Texas, Lotto, etc.). Token issuing makes it easy to implement independent billing models in different Đapps systems. Each token and ZNC anchor according to the deposit, and that formulate a sound & safe value system.

4) A Complete Eco-system Structure: in the computing power market, we need to build multiple ecological elements such as storage, stage transfer, and access pre-access to enable the computing market.

1.3 Prospective Readers

Đapp Developer，Tester

1.4 Terms and Abbreviation

Asset: Everything that record into the blockchain, collectively referred to as assets.

Asset backtracking: The historical changes of specific assets are extracted from the blockchain in chronological order.

Ledger: A chronological listing of blocks of data linked together in a chronological order to ensure that data is not tampered with and forged by distributed ledgers.

2 Blockchain Monitoring Platform

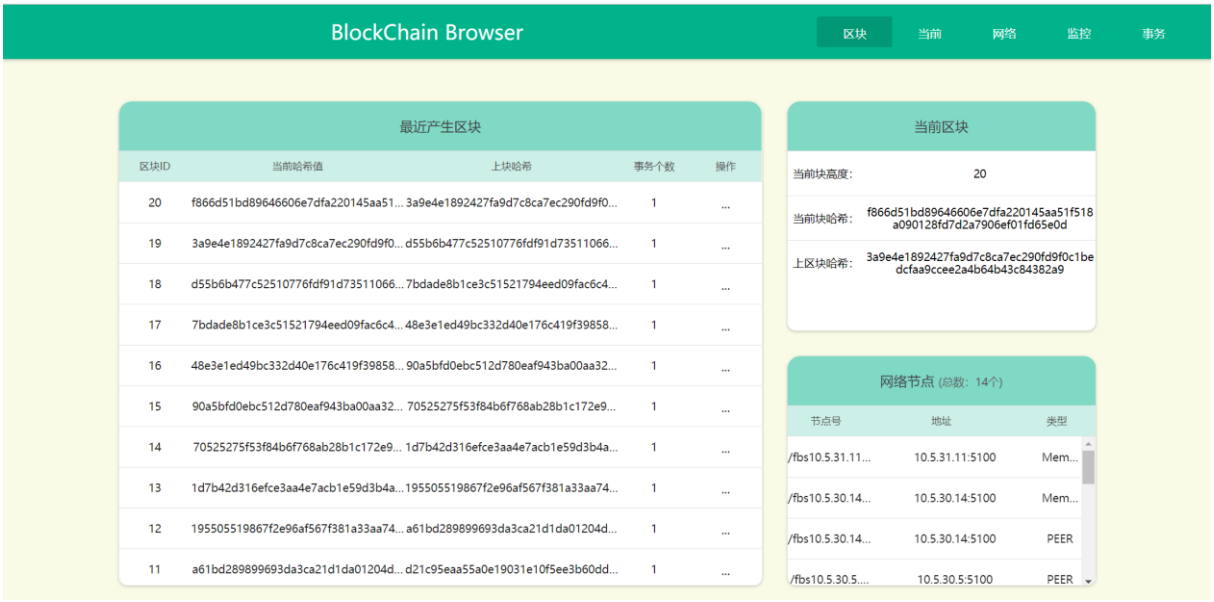
2.1 Description

Blockchain monitoring platform is used to monitor all the data operations in blockchain, and realize the blockchain running status. Such as: monitoring the recent production of the block, the current block, the number of network nodes, the current computing capacity, the average transaction delay, the number of block transactions, recently submitted transaction.

2.2 Address

http://bc.zeusnet.io/prototype/browser/

2.3 Page Screenshot



3 Testing Account Information

The following account information is used for testing the ZeusNetChain prototype functions:

UserID: test

Password: 123456

4 Wallet Interface Description

By the wallet API, the user's own system will connect with ZeusNetChain. Through calling the API, to achieve the assets, data information into the blockchain, query, backtrack and other blockchain operations. ZeusNetChain provide HTTP interface, which includes account login, asset creation, reference file upload, asset information verification, asset transaction and transaction query. The parameters are submitted in post in Json format.

4.1 Account Login

Participants account login the ZeusNet to get the wallet IP.

4.1.1 Request Message

Parameter	Name	Type	Remark	Is Null
orderNo	Order No	String(32)	The unique number of each request	N
userName	User Name	String(64)		N
passwd	Password	String(64)		N
ledgerId	Ledger ID	String(32)		N

Sample:


```
{
    "userName":"jack1",
    "passwd":"1223",
    "orderNo":"10011",
    "ledgerId":"123"
}
```


4.1.2 Response Message

Parameter	Name	Type	Remark	Is Null
orderNo	Order No	String(32)	Same value in request message	N
errorCode	Error Code	String(6)	errorCode=000000 (Six zero) means success, others code mean some error occurred.	N
errorDesc	Error Description	String(256)		N
userId	User ID	String(64)		N
userKey	User's Public Key	String(128)		N

Sample:


```
{
  "errorCode": "000000",
  "errorDesc": "success",
  "userId": "0001110101001",
  "userKey":
  "04005f56800c1c8a2961979f033cfa3507f72a44b25ea6efb25a6392485e55c5b3214ce374374
  18e397d723ce dc52cb219524f2383b45010a3e5aab9ea9304c7fc",
  "orderNo": "10006"
}
```

4.2 Asset Creation

User can submit data into the blockchain by creating assets.

Creating assets requires the asset alias, asset keywords, asset attachment information, and data to be placed in the blockchain.

4.2.1 Request Message

Parameter	Name	Type	Remark	Is Null
orderNo	Order No	String(32)	Unique number per request	N
userKey	Public Key Hash	String(64)		N
userId	User ID	String(64)		N
alias	Asset Alias	String(32)		
dataTable	Data Index Words	String(50)	Multiple index words split by '^'. Example: def^book	Y
filePath	Attachment Files Paths	String（160）	Multiple files paths split by '^'	Y
metadata	Attachment Data	String(1548)		Y
type	Asses Typr	String(2)	Assest Type （0=TOKEN， 1=FBC， 2=BTC， 3=LTC）	N

Sample：

```
{
    "orderNo":"10013",
    "userKey":"11",
    "userId":"0001110101001",
    "alias":"def",
    "dataTable":"123",
    "filePath":"11",
    "type":"1",
    "metadata":"1"
}
```

4.2.2 Response Message

Parameter	Name	Type	Remark	Is Null
orderNo	Order No	String(32)	Same value in request message	N
errorCode	Error Code	String(6)	Return Success, the Error Code is 000000	N
errorDesc	Error Description	String(256)	If return success it will be Null, otherwise will have error description.	N
assetId	Asset ID	String(64)	If return success it will be Asset ID	N

Sample:


```
{
    "orderNo": "10013",
    "errorCode": "000000",
    "errorDesc": "success",
    "assetId": "ff808081000065c5015d7cf740460004"
}
```


4.3 Asset Reference File Upload

Interface for uploading asset reference file. File upload is done via 'form'. You need to follow the URL below these parameters.

4.3.1 Request Message

Parameter	Name	Type	Remark	Is Null
orderId	Order ID	String(32)		N
userId	User ID	String(64)		N
userKey	User's Public Key	String(128)		N

Sample:

orderId=10039& userId=00011101010001& userKey=123

4.3.2 Response Message

Parameter	Name	Type	Remark	Is Null
orderId	Order ID	String(32)		N
errorCode	Error Code	String(6)		N
errorDesc	Error Description	String(256)		N
fileId	File ID	String(64)	Split by '^'	N

Sample:


```
{
    "fileId": "[ff80808100006f43015d7d8e0f500004]",
    "orderId": "10039",
    "errorDesc": "success",
    "errorCode": "000000"
}
```


4.4 Asset Reference File Download

Interface for downloading asset reference file. You need to follow the URL below these parameters.

4.4.1 Request Message

Parameter	Name	Type	Remark	Is Null
orderId	Order ID	String(32)		N
userId	User ID	String(64)		N
fileId	File ID	String(64)		N

Sample:

fileId=ff8080810000705d015d7da9212600001& order Id =10057& userId=0001110101001

4.4.2 Response Message

Parameter	Name	Type	Remark	Is Null
errorCode	Error Code	String(6)		N
errorDesc	Error Description	String(256)		N

Sample:


```
{
    "errorDesc": "000111010100133:  User not exist. ",
    "errorCode": "000001"
}
```


4.5 Asset Query

Query the data in blockchain.

4.5.1 Request Message

Parameter	Name	Type	Remark	Is Null
orderNo	Order No	String(32)	Unique number per request	N
userId	User ID	String(64)		N
userKey	User's Public Key	String(128)	The key got when login success	N
dataTable	Data Index Words	String(50)	Split by '^'	Y
pageNo	Page No	Int		Y
pageSzie	Page Size	Int		Y

Sample:


```
{
    "orderNo": "2000001",
    "userId": "xxxxxxxxxx",
    "userKey": "xxxxxxxxx",
    "dataTable": "Finance^Ticket",
    "pageNo": 1,
    "pageSzie": 20
}
```

4.5.2 Response Message

Parameter	Name	Type	Remark	Is Null
orderNo	Order No	String(32)	Same value in request message	N
errorCode	Error Code	String(6)		N
errorDesc	Error Description	String(256)		N
assetCount	Asset Count	Int	Number of the Asset which meet the query criteria	N
pageNo	Page No	Int		N
assets	Asset list	List<Asset>		Y

Sample:


```
{
    "orderNo": "10021",
    "errorCode": "000000",
    "errorDesc": "success",
    "assetCount": 3,
    "pageNo": 0,
    "assets": [
    {
        "assetId": "ff80808100006de4015d7d2b3a470001",
        "type": "1",
        "alias": "def",
        "dataTable": "123",
        "metadata": "1",
        "amount": 0,
        "count": 0,
        "status": "1",
        "createTime": "2017-07-26 12:32:52",
        "updateTime": "2017-07-26 12:32:52"
    },
    {
        "assetId": "ff8080810000065c5015d7cf740460004",
        "type": "1",
        "alias": "def",
        "dataTable": "123",
        "metadata": "1",
        "filePath": "11",
        "amount": 0,
```

```
            "count": 0,
            "status": "1",
            "createTime": "2017-07-26 11:36:05",
            "updateTime": "2017-07-26 11:36:05"
        },
        {
            "assetId": "ff80808100006128015d7cb9ed1a0002",
            "type": "0",
            "alias": "default",
            "amount": 0,
            "count": 0,
            "status": "1",
            "createTime": "2017-07-26 10:29:06",
            "updateTime": "2017-07-26 10:29:06"
        }
    ]
}
```


4.6 Asset Backtrack

Get back the related history data of the certain asset from the blockchain. When a fault occurs, it can quickly and efficiently locate, thereby reducing the downtime; when the network is attacked, you can quickly locate, analyze and obtain evidence.

4.6.1 Request Message

Parameter	Name	Type	Remark	Is Null
orderId	Order ID	String(32)	Unique number per request	N
userId	User ID	String(64)		N
userKey	User's Public Key	String(128)		N
assetId	Asset_ID	String(64)		N
level	Backtrack Level	Int	Backtrack Level, Max :5	N

Sample:


```
{
    "orderId":"10025",
    "userKey":"11",
    "userId":"0001110101001",
    "assetId":"ff80808100006de4015d7d2b3a470001",
    "level":"1"
```

}

4.6.2 Response Message

Parameter	Name	Type	Remark	Is Null
orderId	Order ID	String(32)	Same value in request message	N
errorCode	Error Code	String(6)		N
errorDesc	Error Description	String(256)		N
transactions	History Data List	List<Transaction>		N

Sample:


```
{
    "orderId": "10025",
    "errorCode": "000000",
    "errorDesc": "success",
    "transactions": [
        {
            "assetId": "ff80808100006de4015d7d2b3a470001",
            "type": "1",
            "alias": "def",
            "dataTable": "123",
            "metadata": "1",
            "amount": 0,
            "count": 0,
            "status": "1",
            "createTime": "2017-07-26 12:32:52",
            "updateTime": "2017-07-26 12:32:52"
        }
    ]
}
```

5 Appendix

5.1 Interface Type

No	Status Name	Status Value
1	Common Interface	01
2	Transaction Interface	02
3	Query Interface	03

5.2 Interface Name

Interface Name	Module Name	Interface Code	Interface Type
UserLogin	user	002	01
AssetCreate	ast	005	02
ReferenceFileUpload	sys	006	03
ReferenceFileDownload	sys	007	04
AssetQuery	ast	008	05
AssetBacktrack	ast	009	06

5.3 Interface Address

Name	Address
UserLogin	/usr/pblin.do?fh=LINUSR0000000J00&resp=bd&bd
CreateLedger	/ldg/pbnew.do?fh=NEWLDG0000000J00&resp=bd&bd
AssetCreate	/ast/pbcrt.do?fh=CRTAST0000000J00&resp=bd&bd
ReferenceFileUpload	/sys/upload.do
ReferenceFileDownload	/sys/download.do
AssetQuery	/ast/pbqry.do?fh=QRYAST0000000J00&resp=bd&bd
AssetBacktrack	/ast/pbtra.do?fh=TRAAST0000000J00&resp=bd

5.4 Asset Fields Description

Name	Description	Type	Remark
assetId	Asset ID	String(64)	
type	Asset Type	String(8)	FBC; BTC; LTC; TOKEN

alias	Asset Alias	String(32)	
dataTable	Data Index Words	String(50)	
metadata	Attachment Data	String(1548)	
filePath	Reference File ID	String(50)	Split by '^'
amount	Transaction Amount in Asset	Decimal(10,4)	
count	Count	Decimal(10,0)	
status	Status	String(12)	0: Invalidity; 1: Create; 2: IntoBlock; 3: Validity
creatTime	Create Time	datetime	
updateTime	Update Time	datetime	

5.5 Transaction Fields Description

Name	Description	Type	Remark
transactionId	Transaction ID	String(64)	
srcAssetId	Source Asset ID	String(8)	
fromUserId	Initial User ID	String(32)	
fromUserName	Initial User Name	String(50)	
destAssetId	Destination Asset ID	String(1548)	
toUserId	Target User ID	String(50)	
toUserName	Target User Name	String(50)	
amount	Transaction Amount in Asset	Decimal(10,4)	
count	Count	Decimal(10,0)	
status	Status	String(10)	0: Invalidity; 1: Create; 2: IntoBlock; 3: Validity
creatTime	Create Time	datetime	
endTime	Update Time	datetime	