EQcoin Bible

Table of contents

1. Terms

1. EQcoin is the original commodity of EQcoin ecosystem. EQcoin is a cryptocurrency，hereinafter referred to as "**EQC**".

2. Type represents the type of lock，hereinafter referred to as "**T**".

3. Operation is defined in Section 2.7.1 below，hereinafter referred to as "**OP**".

2. About EQcoin

2.1 What's EQcoin?

EQcoin is the first Passport oriented decentralized finance ecosystem of the people, by the people, for the people. EQcoin is open source, decentralized, permissionless, distributed and public shared digital ledger. Passport and EQC are the original commodities of EQcoin ecosystem. Passport and EQC are issued and circulated according to the EQcoin consensus mechanism via the EQcoin ecosystem based on the decentralized finance, everyone can participate in the issuance and circulation of Passport and EQC via crowdsourcing(private property is

sacrosanct). The evolution of EQcoin is based on crowdsourcing, everyone can improve and perfect EQcoin via EQcoin Improvement Proposal.

2.2 What's Passport?

Passport is the original commodity of EQcoin ecosystem. Passport has a status, an ID, an EQcoin balance, a nonce, a LockMate and a state root that can send transactions on EQcoin network. The minimum balance value of Passport at the current stage shall not be lower than 51 EQC, and the EQcoin community will decide whether to increase or decrease the minimum balance value of Passport according to the EQcoin Improvement Proposal as needed in the future. Passport ID is a natural number. Passport ID starts from zero and increases one by one according to the order of Passport issuance. Using the status state object, Passport can add or delete Passport relevant state objects created by the EQcoin Improvement Proposal to add or delete its specific functions. Passport can be user controlled and used to deploy multiple smart contracts. Passport owners can provide issue/sell Passport services and smart contract deployment services for everyone and decide how much EQC to charge for issuing/selling Passport and deploying smart contract. Just like Bitcoin Address and Ethereum Account, Passport is anonymous and do not contain information about the owner.

The singularity block is the first block of EQcoin, the No.0 to No.1001 Passports will be issued in this block. Due to “without time at this time“ so singularity block without timestamp.

2.3 What’s EQC?

EQC is the original commodity of EQcoin ecosystem. EQC is a cryptocurrency. EQC needs to be paid in order to use EQcoin decentralized financial services.

2.4 What's Lock?

The Passport owner uses Lock to lock the Passport. Lock has a lock type and the lock relevant state objects. New locks can be created through EQcoin Improvement Proposal to extend the functionality of the lock.

Lock consists of two lock types at the current stage:

2.4.1 T0 lock

T0 lock’s lock type is 0 and use secp256r1(NIST P-256) elliptic curve to lock relevant Passport. T0 lock has a lock type 0, a SHA3-256 public key hash of secp256r1(NIST P-256) elliptic curve and a CRC32C cyclic redundancy checksum.

2.4.2 T1 lock

T1 lock’ s lock type is 1 which is pay to script hash. T1 lock has a lock type 1, a status, a SHA3-256 hash of the relevant redundant lock pairs and a CRC32C cyclic redundancy checksum.

The T1 lock supports the following functions:

1. User is allowed to create a redundant lock pair, which contains N (1<=N<=4) T0 locks from different devices provided by the user. User can select a lock to unlock from the redundant lock pair. Thus, if a single device is damaged, the locks in the redundant devices are still available for use.
2. Single user is allowed to create N(1<=N<=8) lock pairs, and select the M(1<=M<=N) locks in the N lock pairs to unlock.
3. N(1<=N<=8) users are allowed to provide one lock pair per user and must use the N locks in the N lock pairs to unlock.

2.5 What's LockMate?

LockMate has a status, a lock, a publickey that can lock the Passport and store lock relevant state objects. Because the corresponding public key is stored in LockMate when the lock is unlocked for the first time, it can resist preimage attacks.

2.6 Passport and EQC total supply

The total supply of Passport at current stage is 8,388,607. The total supply of Passport may be increased in the future according to the EQcoin Improvement Proposal, and the EQcoin community will determine the total and maximum supply of Passport as needed.

The total supply of EQC is a constant 210,000,000,000 and the decimal is 8. The first block, the Singularity block, will issue 21,000,000 EQCs, and then will issue 21,000,000 EQCs every year.

2.7 What’s Transaction

Transaction is essentially a signed set of instructions from one Passport. Transaction is used to affect a state change on the EQcoin blockchain, such as transfer of funds, change the lock of Passport or execute a function within a smart contract.

Transaction has a status, a Passport ID, a nonce, one or more TxOut arrays and a signature. Using the status state object, Transaction can add or delete Transaction TxOuts created by the EQcoin Improvement Proposal to add or delete its specific functions.

2.7.1 What’s Operation

Operation is essentially a set of instructions from one Passport. Operation is used to affect a state change on the EQcoin blockchain,

such as change Passport ' s lock or deploy a smart contract. Each OperationTxOut contains one or more operation.

Operation has an OP ID and one or more OP state objects. New operations can be created through EQcoin Improvement Proposal to extend the functionality of the operation.

Operation consists of one operation type at the current stage:

1. ChangeLockOP

ChangeLockOP is used to change relevant Passport ' s lock. ChangeLockOP has an OP ID of 0 and a lock which is the new lock for the current Passport.

Transaction consists of four TxOut types at the current stage:

1. ZionTxOut

ZionTxOut is used to issue passports. A maximum of 129 ZionTxOuts can be included in the ZionTxOut array.

2. OperationTxOut

OperationTxOut is used to execute operation, for example ChangeLockOP. A maximum of 65 OperationTxOuts can be included in the OperationTxOut array.

3. TransferTxOut

TransferTxOut is used to transfer EQC. A maximum of 257 TransferTxOuts can be included in the TransferTxOut array.

4. SmartContractTxOut

SmartContractTxOut is used to execute smart contract function. A maximum of 5 SmartContractTxOuts can be included in the SmartContractTxOut array.

2.8 Transaction use case

2.8.1 Issue Passport

1. Adam issues passport and transfers 101 EQCs for Eve (Lock: 0bb...bb).

Before Adam sends the transaction:

Adam's Passport	
Status: 0 ID: 0 Nonce: 0 Balance: 2100000000000000	
LockMate	Lock: 0aa...aa Publickey: null

Transaction sent by Adam to issue Passport:

Transaction
Status[1]: 0000000[1][2]
Passport ID: 0
Nonce: 0

[1] The type of the Status state object is EQCBits.

[2] Indicates whether transaction includes ZionTxOut, 0: excludes, 1: includes.

ZionTxOut	Status[3]: 0000000[4]0[5]
	Lock: bb...bb[6] Value: 10100000000
Signature: xx...xx	

The size of the transaction is 105 bytes.

After Adam sends the transaction:

Adam's Passport	
Status: 0 ID: 0 Nonce: 1 Balance: 2099989800090000	
LockMate	Lock: 0aa...aa Publickey: xx...xx

Eve's Passport
Status: 0 ID: 1

---

[3] The type of the Status state object is EQCBits.

[4] This state object includes a series of consecutive bits. When ZionTxOut includes only one sub-ZionTxOut uses it to record the lock type part of the current sub-ZionTxOut' s lock, and when ZionTxOut includes multiple sub-ZionTxOuts uses it to record the number of sub-ZionTxOuts. The current record value is the lock type part 0000000(0) of the sub-ZionTxOut' s lock.

[5] Indicates whether ZionTxOut includes multiple sub-ZionTxOuts, 0: one, 1: multiple.

[6] When ZionTxOut includes only one sub-ZionTxOut uses it to record the hash part of the lock, and when ZionTxOut includes multiple sub-ZionTxOuts uses it to record the full lock. The current record value is the public key hash part bb...bb of sub-ZionTxOut' s T0 lock.

Nonce: 0 Balance: 10100000000	
LockMate	Lock: 0bb...bb Publickey: null

2. Adam issues passports and transfers 101 EQCs for Moses (Lock: 0cc...cc) and Noah (Lock: 0dd...dd) and charge the service fee of 1 EQC per person. Therefore, after deducting the service fee, the transfer amount is 100 EQC.

Before Adam sends the transaction:

Adam's Passport	
Status: 0 ID: 0 Nonce: 1 Balance: 2099989800090000	
LockMate	Lock: 0aa...aa Publickey: xx...xx

Transaction sent by Adam to issue Passports:

Transaction
Status: 00000001
Passport ID: 0
Nonce: 1

ZionTxOut	Status：0000000[7]1[8]
	Lock: 0cc...cc Value: 10000000000
	Lock: 0dd...dd Value: 10000000000
Signature: xx...xx	

The size of the transaction is 144 bytes.

After Adam sends the transaction：

Adam's Passport	
Status: 0 ID: 0 Nonce: 2 Balance: 2099969800080000	
LockMate	Lock: 0aa...aa Publickey: xx...xx

Moses' Passport
Status: 0 ID: 2 Nonce: 0

[7] Indicates ZionTxOut includes multiple sub-ZionTxOuts.

[8] Record the current number of sub-ZionTxOuts is 0000000(2).

Balance: 10000000000	
LockMate	Lock: 0cc...cc Publickey: null

Noah's Passport	
Status: 0 ID: 3 Nonce: 0 Balance: 10000000000	
LockMate	Lock: 0dd...dd Publickey: null

2.8.2 Transfer

1. Adam transfers 101 EQCs to Moses.

Before Adam sends the transaction：

Adam's Passport	
Status: 0 ID: 0 Nonce: 2 Balance: 2099969800080000	
LockMate	Lock: 0aa...aa

	Publickey: xx...xx

Transaction sent by Adam to transfer:

Transaction	
Status: 0000[9]1[10]000	
Passport ID: 0	
Nonce: 2	
TransferTxOut[11]	Passport ID: 2 Value: 10100000000
Signature: xx...xx	

The size of the transaction is 72 bytes.

After Adam sends the transaction:

Adam's Passport	
Status: 0 ID: 0 Nonce: 3 Balance: 2099959700070000	
LockMate	Lock: 0aa...aa Publickey: xx...xx

---

[9] When transaction includes TransferTxOut indicates whether TransferTxOut includes multiple sub-TransferTxOuts, 0: one, 1: multiple, otherwise indicates whether transaction includes SmartContractTxOut, 0: excludes, 1: includes.

[10] Indicates whether transaction includes TransferTxOut, 0: excludes, 1: includes.

[11] EQcoin uses EQCHelix to store the transfer value and relevant Passport ID's bytes' length in TransferTxOut. On the underlying storage, Value is stored first, followed by Passport ID.

Moses' Passport	
Status: 0 ID: 2 Nonce: 0 Balance: 20100000000	
LockMate	Lock: 0cc...cc Publickey: null

3. Adam transfers 101 EQCs to Moses and Noah.

Before Adam sends the transaction:

Adam's Passport	
Status: 0 ID: 0 Nonce: 3 Balance: 2099959700070000	
LockMate	Lock: 0aa...aa Publickey: xx...xx

Transaction sent by Adam to transfer:

Transaction
Status: 0001[12]1000
Passport ID: 0

[12] Indicates TransferTxOut includes multiple sub-TransferTxOuts.

Nonce: 3	
TransferTxOut	Array length: 00000000[13]
	Passport ID: 2 Value: 10100000000
	Passport ID: 3 Value: 10100000000
Signature: xx...xx	

The size of the transaction is 78 bytes.

After Adam sends the transaction:

Adam's Passport	
Status: 0 ID: 0 Nonce: 4 Balance: 2099939500060000	
LockMate	Lock: 0aa...aa Publickey: xx...xx

Moses' Passport
Status: 0 ID: 2 Nonce: 0

[13] Record the current number of sub-TransferTxOuts is 00000000(2).

Balance: 30200000000	
LockMate	Lock: 0cc...cc Publickey: null

Noah's Passport	
Status: 0 ID: 3 Nonce: 0 Balance: 20100000000	
LockMate	Lock: 0dd...dd Publickey: null

2.8.3 Change lock

1. Adam changes his Passport’ s lock to 0bb...bb.

Before Adam sends the transaction：

Adam's Passport	
Status: 0 ID: 0 Nonce: 4 Balance: 2099939500060000	
LockMate	Lock: 0aa...aa

	Publickey: xx...xx

Transaction sent by Adam to change lock:

Transaction	
Status: 000000[1][14]0	
Passport ID: 0	
Nonce: 4	
OPTxOut	Status[15]: 0000000[16]0[17] Lock: 0bb...bb[18]
Signature: xx...xx	

The size of the transaction is 101 bytes.

After Adam sends the transaction:

Adam's Passport
Status: 0 ID: 0 Nonce: 5 Balance: 2099939500050000

---

[14] Indicates whether transaction includes OPTxOut, 0: excludes, 1: includes.

[15] The type of the Status state object is EQCBits.

[16] This state object includes a series of consecutive bits. When OPTxOut includes only one sub-OPTxOut uses it to record the OP ID part of the sub-OPTxOut, and when OPTxOut includes multiple sub-OPTxOuts uses it to record the number of OPTxOuts. The current record value is the OP ID part 0000000(0) of the ChangeLockOP.

[17] Indicates whether OPTxOut includes multiple sub-OPTxOuts, 0: one, 1: multiple.

[18] When OPTxOut includes only one sub-OPTxOut uses it to record the OP body part of the OP, and when OPTxOut includes multiple sub-OPTxOuts uses it to record the full OP. The current record value is the lock part 0bb...bb of ChangeLockOP.

LockMate	Lock: 0bb...bb Publickey: null

2.8.4 Execute smart contract

1. Adam executes the Buy function (ID: 4) of the EQswap smart contract (ID: 1002.2) and uses 0.00000051 EQC to buy 201 Bethard tokens from Bethard.

Before Adam sends the transaction:

Adam's Passport	
Status: 0 ID: 0 Nonce: 5 Balance: 2099939500050000	
LockMate	Lock: 0bb...bb Publickey: null

Transaction sent by Adam to execute smart contract:

Transaction
Status: 000[19]1[20]0000
Passport ID: 0

[19] When transaction includes SmartContractTxOut indicates whether SmartContractTxOut includes multiple sub-SmartContractTxOuts, 0: one, 1: multiple, otherwise reserved for indicates other states of the transaction.

[20] Indicates whether transaction includes SmartContractTxOut, 0: excludes, 1: includes.

Nonce: 5	
SmartContractTxOut	Status: 00100[21]01[22]0[23] Smart contract ID: 1002.2 Function ID: 4[24] Value: 51
Signature: xx...xx	

The size of the transaction is 72 bytes.

After Adam sends the transaction:

Adam's Passport	
Status: 0 ID: 0 Nonce: 6 Balance: 2099939500039949	
Bethard token: 20100000000	
LockMate	Lock: 0bb...bb Publickey: xx...xx

2. Adam executes the Buy function (ID: 4) of the EQswap smart contract (ID: 1002.2) and uses 0.00000051 EQC to buy 201 Bethard tokens from

[21] When the current smart contract function is called once uses it to record the function ID, and when the current smart contract function is called multiple times uses it to record the number of calls. The current record value is the current smart contract function ID 00100(4) which is Buy function.

[22] Record the bytes of the Passport ID in the smart contract ID. The current record value is the size of the byte stream corresponding to Passport ID 1002, which is 01(2) bytes.

[23] Indicates whether current smart contract function includes multiple calls, 0: one, 1: multiple.

[24] The value is saved in the Status state object identified by note 21.

Bethard, then executes the betting function (ID: 3) of the Bethard horse racing smart contract (ID: 1002.3) and uses 201 EQCs to bet that No. 9 of the Royal Ascot's Her Majesty's Plate will win.

Before Adam sends the transaction:

Adam's Passport	
Status: 0 ID: 0 Nonce: 6 Balance: 2099939500039949	
Bethard token: 20100000000	
LockMate	Lock: 0bb...bb Publickey: xx...xx

Transaction sent by Adam to execute smart contract:

Transaction	
Status: 001[25]10000	
Passport ID: 0	
Nonce: 6	
	Status: 000000[26]00[27]
SmartContractTxOut	Status: 00100010 Smart contract ID: 1002.2

[25] Indicates SmartContractTxOut includes multiple sub-SmartContractTxOuts.

[26] Reserved status flag bits.

[27] Record the number of sub-SmartContractTxOuts. The current record value is 00(2).

	Function ID: 4 Value: 51
	Status: 00011010 Smart contract ID: 1002.3 Function ID: 3 Value: 20100000000 Winner: 9
Signature: xx...xx	

The size of the transaction is 83 bytes.

After Adam sends the transaction:

Adam's Passport	
Status: 0 ID: 0 Nonce: 7 Balance: 2099919400029898	
Bethard token: 40200000000	
LockMate	Lock: 0bb...bb Publickey: xx...xx

3. Adam executes the Transfer function (ID: 1) of the Bethard token contract (ID: 1002.1) and transfer 100 Bethard tokens to Eve, Moses and Noah.

Before Adam sends the transaction:

Adam's Passport	
Status: 0 ID: 0 Nonce: 7 Balance: 2099919400029898	
Bethard token：40200000000	
LockMate	Lock: 0bb...bb Publickey: xx...xx

Transaction sent by Adam to execute smart contract：

Transaction	
Status: 00100000	
Passport ID: 0	
Nonce: 5	
SmartContractTxOut	Status: 00001[28]011[29] Smart contract ID: 1002.1 Function ID: 1 Passport ID: 1 Value: 100 Passport ID: 2 Value: 100

[28] The current record value is the number of current smart contract function calls: 00001(3).

[29] Indicates current smart contract function includes multiple calls.

	Passport ID: 3 Value: 100
Signature: xx...xx	

The size of the transaction is 77 bytes.

After Adam sends the transaction:

Adam's Passport	
Status: 0 ID: 0 Nonce: 8 Balance: 2099919400019898	
Bethard token: 10200000000	
LockMate	Lock: 0bb...bb Publickey: xx...xx

Eve's Passport	
Status: 0 ID: 1 Nonce: 0 Balance: 10100000000	
Bethard token: 10000000000	
LockMate	Lock: 0bb...bb

	Publickey: null

Moses' Passport	
Status: 0 ID: 2 Nonce: 0 Balance: 30200000000	
Bethard token : 10000000000	
LockMate	Lock: 0cc...cc Publickey: null

Noah's Passport	
Status: 0 ID: 3 Nonce: 0 Balance: 20100000000	
Bethard token : 10000000000	
LockMate	Lock: 0dd...dd Publickey: null

2.8.5 Complex transaction

Adam issues passport and transfers 51 EQC for Amon （Lock: 0ee...ee, transfers 101 EQCs to Moses, executes the Buy function (ID: 4) of the EQswap smart contract (ID: 1002.2) and uses 0.00000051 EQC to buy 201 Bethard tokens from Bethard and changes his Passport’ s lock to 0ff...ff and set the power price to 11 to execute transactions at a faster accounting rate.

Before Adam sends the transaction：

Adam's Passport	
Status: 0 ID: 0 Nonce: 8 Balance: 2099919400019898	
Bethard token：10200000000	
LockMate	Lock: 0bb...bb Publickey: xx...xx

Complex Transaction sent by Adam：

Transaction
Status: 00101[1][30]11
Passport ID: 0

[30] Indicates whether transaction specifies a custom PowerPrice, 0: default, 1: custom.

Nonce: 8	
ZionTxOut	Status: 00000000 Lock: ee...ee Value: 5100000000
OPTxOut	Status: 00000000 Lock: 0ff...ff
PowerPrice	Value: 11
TransferTxOut	Passport ID: 2 Value: 10100000000
SmartContractTxOut	Status: 00100010
	Smart contract ID: 1002.1 Function ID：4 Value: 51
Signature: xx...xx	

The size of the transaction is 151 bytes.

After Adam sends the transaction：

Adam's Passport
Status: 0 ID: 0 Nonce: 9 Balance: 2099904200008847
Bethard token：30300000000

LockMate	Lock: 0ff...ff Publickey: null

Moses' Passport	
Status: 0 ID: 2 Nonce: 0 Balance: 40300000000	
Bethard token: 10000000000	
LockMate	Lock: 0cc...cc Publickey: null

Amon's Passport	
Status: 0 ID: 4 Nonce: 0 Balance: 5100000000	
LockMate	Lock: 0ee...ee Publickey: null

2.9 What's ZeroOneMerklePatriciaTrie

ZeroOneMerklePatriciaTrie is used to store the Global State of the Passport、 Transaction and relevant state objects(for example Smart Contract relevant state objects)in each block of EQcoin. Each Global State object has a unique ID, which is a natural number encoded consecutively from zero. In the ZeroOneMerklePatriciaTrie, it is bit by bit addressed from high bit to low bit according to the binary value of the ID of the relevant state object. ZeroOneMerklePatriciaTrie includes only two keys, 0 and 1.

2.9.1 What’s Node

Node is the key node that constitutes the ZeroOneMerklePatriciaTrie dictionary tree. Node has a status, a key, a ZeroNode, a OneNode and a value. ZeroOneMerklePatriciaTrie includes only two nodes, ZeroNode and OneNode. Node's underlying storage implementation includes a HashKey(Hash of Node's binary raw data, used to support state object verification based on light client protocol) and a StorageKey(UpdateNonce(A natural number starting from 0 and increments by 1 with each modification of the node) of Node, used to access state objects from StateDB).

2.9.1.1 What’ s Status

Status is used to identify the composition of the state objects included in the Node. The type of the Status state object is EQCBits.
The default order of state objects that Node includes is: ZeroNode, OneNode, and Value.
Status identifies bit resolution:
1. The default universal identifier bit of the Node.
00000$0^{31}0^{32}0^{33}$
2. When Node includes two state objects(ZeroNode&Value or OneNode&Value).

000$0^{34}0^{35}$000

Note: The state object has the smallest StorageKey hereinafter referred to as “A”(when there are multiple equal value minimums, the one with the lowest default order is taken), another state object hereinafter referred to as “B”. If A’ s StorageKey equals B’ s StorageKey, the underlying data is A's StorageKey . If A’ s StorageKey is not equal to B’ s StorageKey, the underlying data is A's StorageKey and (B's StorageKey - A's StorageKey)(this saves more storage space than directly stores the two state objects’ StorageKey). For example, if B's StorageKey is 100,001 and the A's StorageKey is 100000, the underlying stored data is 100,000 and 1(this saves a lot of storage space than direct storage 100,000 and

---

[31] Indicates whether node includes OneNode, 0: excludes, 1: includes.
[32] Indicates whether node includes ZeroNode, 0: excludes, 1: includes.
[33] Indicates whether node includes Value, 0: excludes, 1: includes.
[34] Identifies whether the two state objects’ StorageKey are equal, 0: no, 1: yes.
[35] Identifies which node state object’ StorageKey is the smallest in the default order, 0: left, 1: right.

100,001). When need to restore the B's StorageKey, can obtain its value through 100000+1.

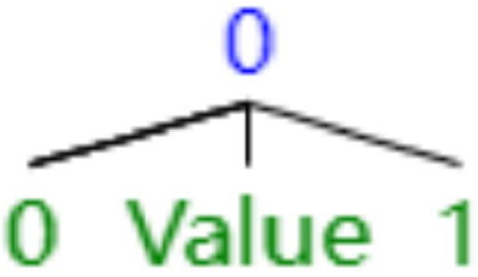
3. When Node includes three state objects(ZeroNode, OneNode and Value).

0[36]0[37]0[38]00[39]000

Note: The state object has the smallest StorageKey hereinafter referred to as "A"(when there are multiple equal minimum values, the one with the lowest default order is taken), the state object after A in the default order hereinafter referred to as "B"(If A is Value, the order is calculated from the beginning, so B is ZeroNode) , the state object after B in the default order hereinafter referred to as "C"(If B is Value, the order is calculated from the beginning, so C is ZeroNode) . The one with the larger StorageKey in B and C hereinafter referred to as "D" and the The one with the smaller StorageKey in B and C hereinafter referred to as "E".

2.9.1.2 What's ZeroNode

As shown in the figure below, the key of ZeroNode is 0.



[36] Identifies whether E's StorageKey is equal to D's StorageKey, 0: no, 1: yes.

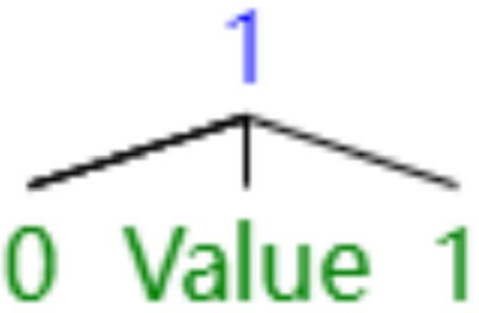
[37] Identifies whether D's StorageKey is equal to A's StorageKey, 0: no, 1: yes.

[38] Indicates B's and C's StorageKey who is bigger, 0: B<=C, 1: B>C.

[39] Indicates which state object has the smallest UpdateNonce, 0: ZeroNode, 1: OneNode, 2: Value.

2.9.1.3 What’ s OneNode

As shown in the figure below, the key of OneNode is 1.

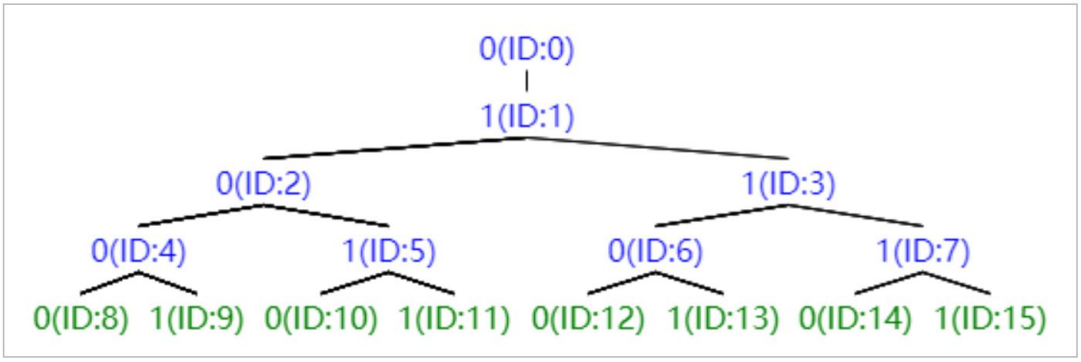


2.9.1.3 What’ s RootNode

RootNode is the root of ZeroOneMerklePatriciaTrie which has not Key and Value but has ZeroNode and OneNode.

2.9.2 Passport/Transaction Global State chart

The location of the Passport/Transaction ID from 0 to 15 is depicted in the following figure.

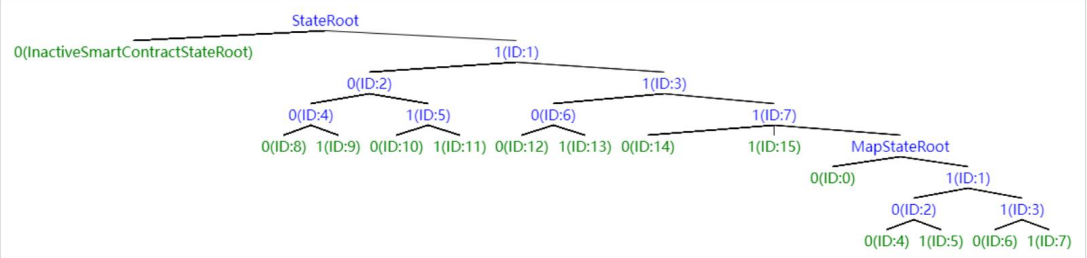


Note: The value of the node in the above figure is the value of the

relevant Passport/Transaction that has been omitted in this figure.

2.9.3 Smart Contract state object Global State chart

The location of the Smart Contract state object ID from 0 to 15 is depicted in the following figure.



2.10 EQcoin roadmap

Stage 1 - Inception

EQcoin mainnet online.

Stage 2 - Era

EQcoin supports smart contracts compatible with the Ethereum EVM.

Stage 3 - New dawn

EQcoin supports cross chain through the Interchain Communication Protocol.

Stage 4 - Nirvana

EQcoin moves from POW to POS.

2.11 EQcoin milestone

2018-01-01 EQcoin officially launched.

2018-04-10 GitHub Initial commit.

2019 Establish an EQcoin test network to achieve multiple miner nodes based on POW consensus mechanisms to mine, send transactions(issue Passport, transfer EQC and change lock), verify blocks, and compete the longest blockchain.

2020-02-10 Register the domain name of www.eqcoin.org.

2021-02-12 Create EQcoin organization in GitHub.

2021-04 Create EQcoin twitter.

2021-04 Create EQcoin facebook.

At present, the overall design of the Inception phase of EQcoin has been completed. We have written thousands of pages of research and development technology documents and the code is about 80% complete and including a total of 33000+ lines. Our developer community currently has 13 members.

2.12 EQcoin GitHub

https://github.com/EQcoin

2.13 Our developer community

Currently we have 13 members. You can visit https://github.com/orgs/EQcoin/people to learn more about our

developer community.

2.14 Copyright

The copyright of all works released by Wandering Earth Corporation or jointly released by Wandering Earth Corporation with cooperative partners are owned by Wandering Earth Corporation and entitled to protection available from copyright law by country as well as international conventions.

Attribution — You must give appropriate credit, provide a link to the license.

Non Commercial — You may not use the material for commercial purposes.

No Derivatives — If you remix, transform, or build upon the material, you may not distribute the modified material.

Wandering Earth Corporation reserves any and all current and future rights, titles and interests in any and all intellectual property rights of Wandering Earth Corporation including but not limited to discoveries, ideas, marks, concepts, methods, formulas, processes, codes, software, inventions, compositions, techniques, information and data, whether or not protectable in trademark, copyrightable or patentable, and any trademarks, copyrights or patents based thereon.

For the use of any and all intellectual property rights of Wandering

Earth Corporation without prior written permission, Wandering Earth Corporation reserves all rights to take any legal action and pursue any rights or remedies under applicable law.