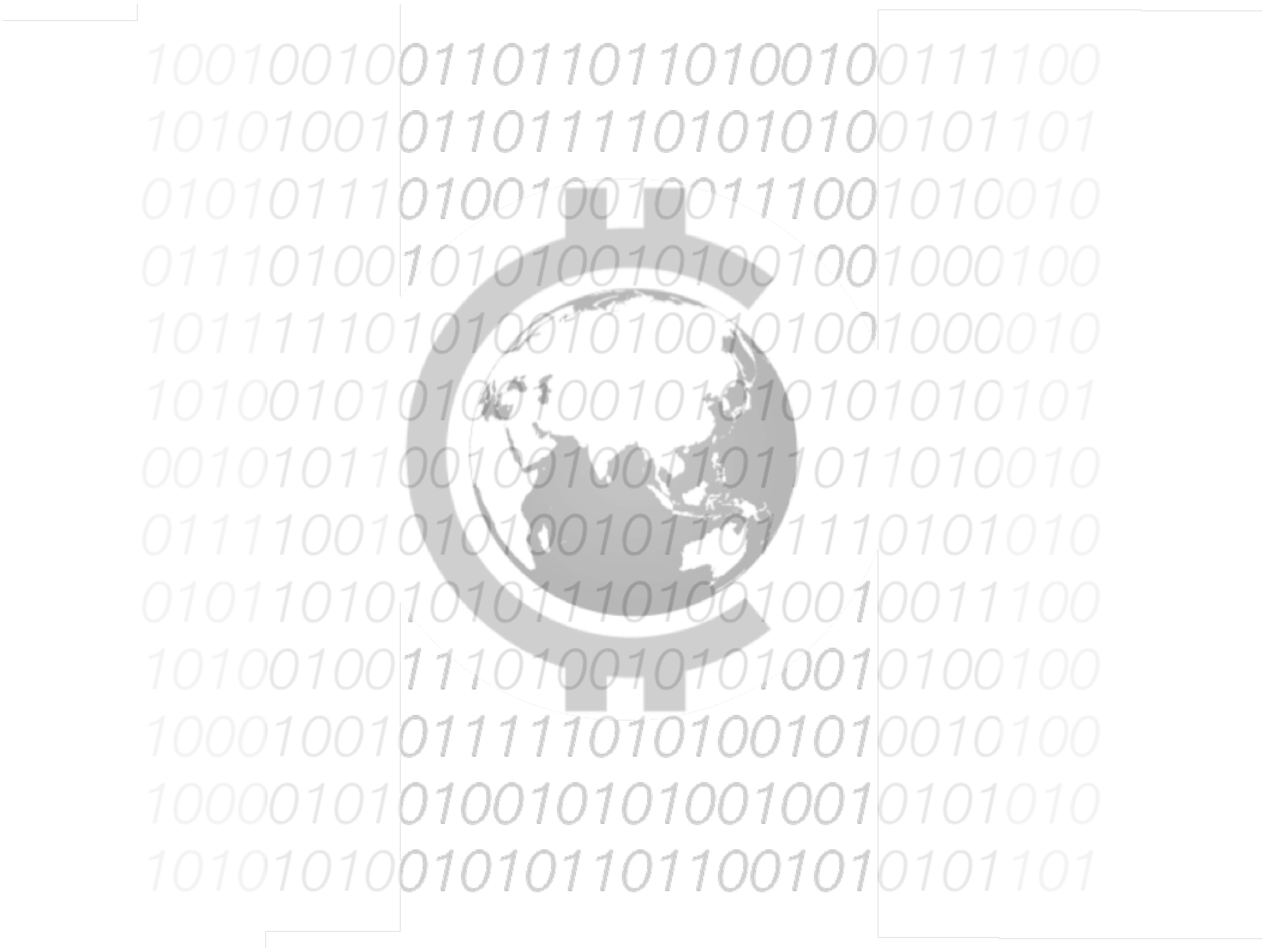
CORION Platform

DEVELOPER GUIDE and TECHNICAL REVIEW
v0.4

www.corion.io

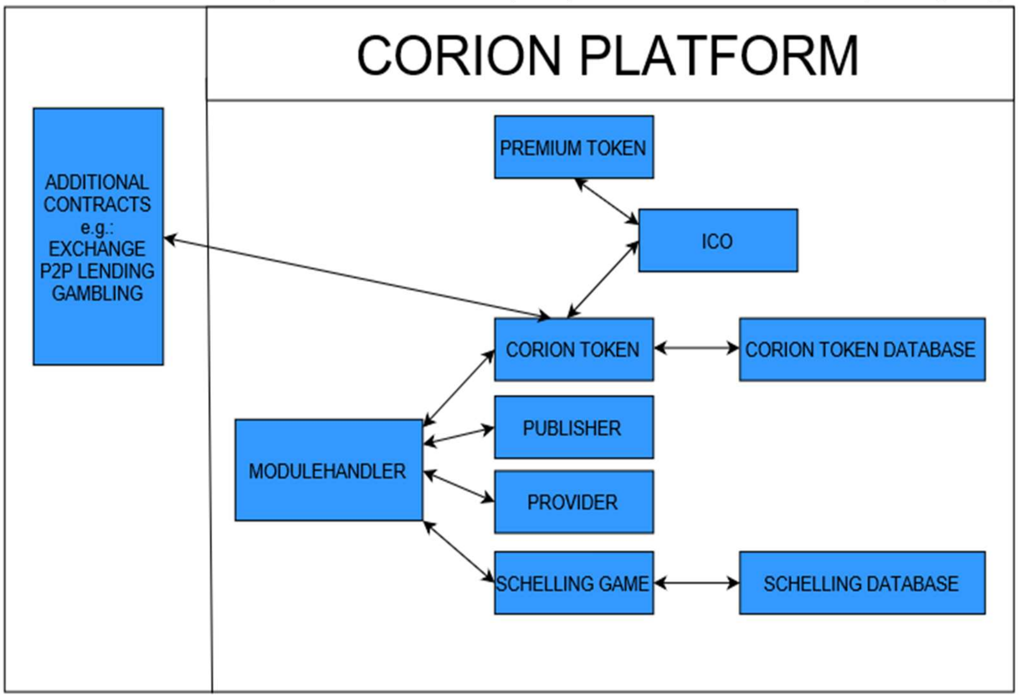


Zug, Switzerland
02.05.2017

Authors: CORION Team: by Attila Dancs attila@corion.io and iFA Rajci Andor ifa@corion.io

www.corion.io

CORION Platform is hosted on Ethereum Classic blockchain (https://en.wikipedia.org/wiki/Ethereum_Classic).
Initially, it consists of 9+1 separate contracts, implemented in Solidity language.



Ethereum Classic blocktime is on average 15 sec. 1 CORION segment = 40.320 blocks = ~1 week.

I. INSTALLING CORION PLATFORM ON BLOCKCHAIN

1. Installing Modulehandler;

2. Installing ICO contract and parameters;

function ico(address foundation, address priceSet, uint256 exchangeRate, uint256

- Address, capable to change the rate
- Rate (Usd, 10000x (eg: 25000 = 2,5Usd))
- Block-height of the ICO start

3. CORION Token database;

4. Installing CORION Token contract and parameters;

Function Token (address _moduleHandler, address _db, address _ExchangeAddress, address _icoAddr, address[] genesisAddr, uint256[] genesisValue) payable {}, where:

- Address Modulehandler
- Address Token database
- Address Exchange
- Address ICO contract
- Array: Setting address of genesis users

e.g.:"0x6e5fd1741e45c966a76a077af9132627c07b0dc1", "0xbd3c601b59f46cc59be3446ba29c66b9182a70b6"]

- Array: Setting balance of genesis users

e.g.: ["1000000000000","1000000"]

this address: 0x6e5fd1741e45c966a76a077af9132627c07b0dc1, get 1000000.000000 CORION Token.

And this address: 0xbd3c601b59f46cc59be3446ba29c66b9182a70b6, get 1.000000

CORION Token.
Addresses [n1,n2,n3...n]
Balances [n1,n2,n3...n]

5. **Installing PREMIUM Token contract and parameters;**

function PREMIUM(address _icoAddr, address[] genesisAddr, uint256[] genesisValue) {}, where:

- Address ICO contract
- Array: Setting address of genesis users

e.g.:"0x6e5fd1741e45c966a76a077af9132627c07b0dc1", "0xbd3c601b59f46cc59be3446ba29c66b9182a70b6"]

- Array: Setting balance of genesis users

e.g.: ["1000","10"] this address:
get 0x6e5fd1741e45c966a76a077af9132627c07b0dc1, 1000 PREMIUM Token.
And this address: 0xbd3c601b59f46cc59be3446ba29c66b9182a70b6, get 10 PREMIUM Token.
Addresses [n1,n2,n3...n]
Balances [n1,n2,n3...n]

6. **Publisher;**

function publisher(address _moduleHandler) {}, where:

- Address Modulehandler

7. **Provider;**

function provider(address _moduleHandler) {}, where:

- Address Modulehandler

8. **Schelling database;**
9. **Schelling;**

function schelling(address _moduleHandler, address _db) {}, where:

- Address Modulehandler
- Address Schelling database

These modules/contracts will be installed on the blockchain, and will be executed by Modulehandler Load function.

II. CORION PLATFORM INITIALIZATION

function load(address foundation, bool forReplace, address Token, address publisher, address Schelling, address Provider) {}
Calling Modulehandler Load functions by these parameters

- Address Foundation
- False (it's not a module change)
- Address CORION Token module
- Address Publisher module
- Address Schelling module
- Address Provider module

calls ICO connect Tokens function, and connects CORION Token and PREMIUM Token with ICO module.
function connectTokens(address _TokenAddr, address _PREMIUMAddr) external {}, where:

- Address CORION Token
- Address PREMIUM Token

III. PROCESS OF MODULE CHANGE

Only the Publisher Contract is allowed to change a module. Module change will be announced here, and can be vetoed or not. Each announcement lasts one week at least, after that the results will be shared.

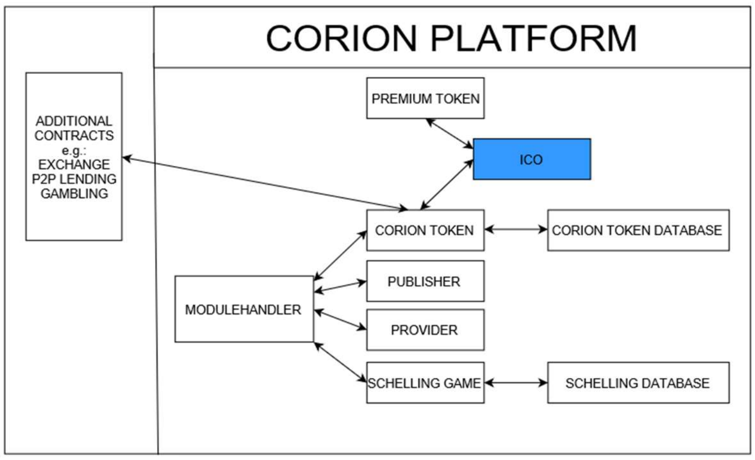
Process of module change:

- Installing new module on blockchain
- Announcement (Publisher contract)
- At least 1 week veto period
- Closing the announcement
- In case of success the new module will replace the old one

IV. PROCESS OF MODULEHANDLER

- Installing Modulehandler on blockchain
- Calling Modulehandler Load function (to Replace must be TRUE)
- Announcement (Publisher contract)
- At least 1 week veto period
- Closing the announcement
- In case of success the new Modulehandler will replace the old one

V. ICO MODULE



Initial Coin Offering will be managed only by the ICO Contract.

- ICO Starts with the initial block-height
- ICO lasts 5 segments
- ICO can be 1 segment/decision

One segment contains 40.320 blocks.
ICO can be closed only after the end of the last segment.
The minimum buy during the ICO is 5 CORION Tokens.

ICO invalidation:

ICO can be invalidated (only during ICO). In this case the users can recall 90% of their deposit. The remaining 10% will be addressed to the Foundation, for marketing costs.
foundationAddress.send(_value * 10 / 100)

If the user's address doesn't have at least 0,2 ETC, this amount will be subtracted from the purchase of CORION Tokens, and this 0,2 ETC will be transferred to the address.

USD/ETC rate during ICO:
The rate is till 125 blocks valid, after that there will be a new rate till the next 125 bocks. The rate will be declared and uploaded by an automated script. In case there are no new rates, the last used rate is valid.

Counting CORION Token during ICO:
reward = value * 1e6 * icoExchangeRate / icoExchangeRateM / 1 Ether;
x = (value * 1e6 * USD_ETC_Rate / 1e4 / 1e18) * Bonus_Percent
e.g.: 2.700000 Token = (1e18 * 1e6 * 22500 / 1e4 / 1e18) * 1.20

Genesis block:
This block will hold the addresses and amounts of CORION Tokens and PREMIUM Tokens (of Pre-ICO buyers, Contracts, Modules) defined.
Address and amount settings will depend on the purchased CORION Tokens during Pre-ICO.

ICO bonuses:
From the ICO starter block-height:

+5760	above block +20% (about 6 day)
+40320	above block +15% (about 1 week)
+80640	above block +10% (about 1 week)
+120960	above block +5% (about 1 week)
+161280	above block +3% till ICO close

For Bonus calculation, it is always the actual block-height that counts.

ICO Affiliate system
The referrer is entitled to a commission, based on the payments of the person referred.
It is automated in the official CORION Wallet, in 3rd party wallet it is self managed.

<1 000 COR		1% upon the bought amount
1 000 COR	=<	2% upon the bought amount
10 000 COR	=<	3% upon the bought amount
100 000 COR	=<	4% upon the bought amount
1 000 000 COR	=<	5% upon the bought amount

e.g.:
1. "A" refers "B".
2. "B" buys 5 000 CORION Tokens -> "A" gets 100 CORION Tokens.
3. "B" buys further 6 000 CORION Token -> "A" gets additional 230 CORION Tokens ((11 000×3%)-100)=230.

- Calling the GetICOReward function is free
- Requirement to be a referrer is to have a balance > 0.000000 CORION Tokens

Participation in ICO:
ICO has a separate Contract. There are 3 ways to participate:
- Through the official CORION Wallet calling "CORION Token buy" function
- Sending ETC to the Contract address. In this case the sender's address will get his CORION (and PREMIUM Token).
- The buy function of the Contract must be called by ETC and parameters, which are:
    1. Beneficiaries public address
    2. Referrals public address

PREMIUM Token during ICO:
PREMIUM Tokens will be generated only during the ICO period.
For every 5000 CORION Tokens bought, the buyer gets one PREMIUM Token:

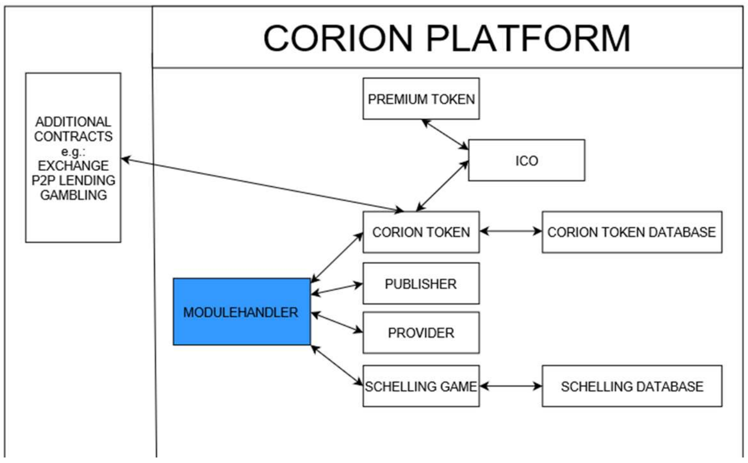
- Buy can be executed in instalments (1x1500, 1x 2000, 1x2000 -> 1 PREMIUM Token)
- Bonus is included
- Affiliate commission excluded
- CORION Token emission excluded
- The generated, but undistributed PREMIUM Tokens will be sent to the Foundation's address.

Operation of CORION Platform during ICO

- Only genesis addresses can transfer CORION Tokens and PREMIUM Tokens to another genesis address
- Providers can be created only by the Foundation
- Schelling game will be automated, participation will be disallowed, CORION Token emission is constant 0,025%/Schelling game, will be 100% distributed among the joined users.
- Affiliate system with these functions works only during the ICO

ICO closing:
Only after the ICO period. After the end of ICO the amount of CORION Tokens will increase by 96%, created directly on the address of Foundation. The ETC on the ICO Contract will be sent to the address of the Foundation.

VI. MODULEHANDLER



This module is the link between the other modules. The function will be called through this module. There are 4 modules to call:

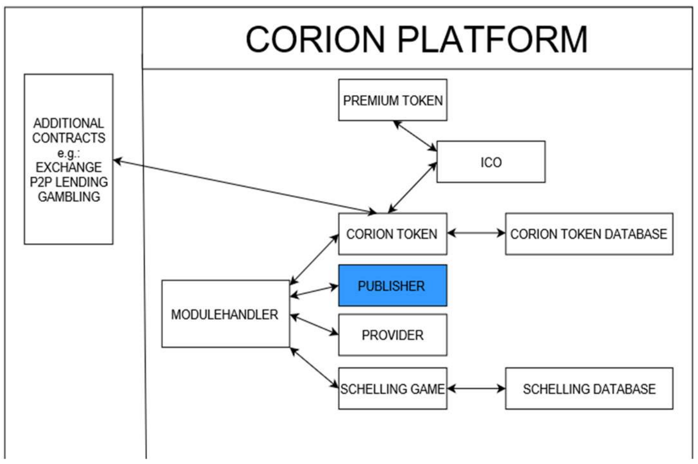
- CORION Token
- Publisher
- Schelling
- Provider

Modulehandler contains 2 further functions:

- Load
- Freeze

Freeze: The address which installed the Modulehandler has the rights to freeze the module minimum for 40320 blocks (1 week), maximum forever. During the frozen zone neither of the modules can be used.

VII. PUBLISHER



This contract is responsible for the announcements of the changes in the modules of the Platform, which can be vetoed by the users.
In this module a White List with users/addresses will be managed, and only these users can start/close/revoke announcements through this Publisher module. The White-List can be managed only by the Owner.
Two main types of announcements:

- Opposable, can be vetoed.
- Non-opposable, cannot be vetoed.

Measuring the veto:
Every CORION Token is a vote. The end of the veto period, the votes will be counted:
CORION Token amount*oppositeRate/100
OppositeRate = The CORION Token amount on the address of the veto
To successfully veto an announcement, there must be minimum 33% votes of the whole CORION Token quantity.

- Vetos >= 33% - Change won't be realized
- Vetos < 33% - Change will be realized

A veto does not entail any transaction fee.

Announcements parameters:

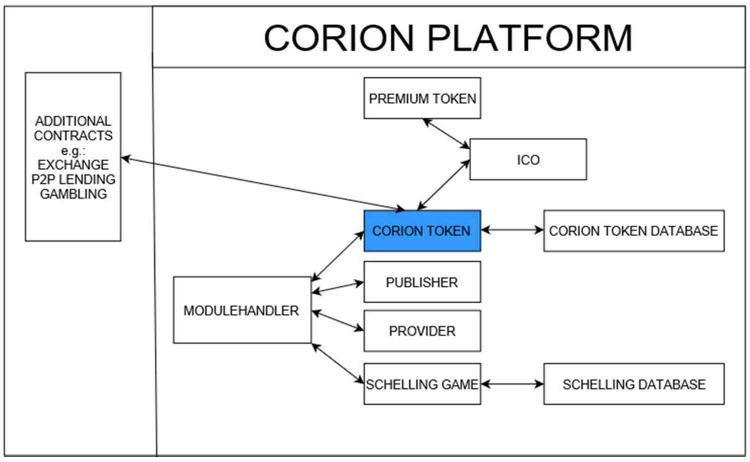
@id	ID
@Type	Title
@Start	Block-height
@End	Planned end of veto period
@Closed	Closed or not
@Announcement	Content
@Link	URL
@Opposited	Status: Vetoed or not
@_str	Field for text
@_uint	Field for numbers
@_addr	Field of address

Announcements can be about:

- newModule: New module connection
- dropModule: Module disconnection
- replaceModule: Module replace

- question: Question towards the community
- transactionFeeRate: Transaction fee rate
- transactionFeeMin: minimum of the transaction fee
- transactionFeeMax: maximum of the transaction fee
- transactionFeeBurn: Burning transaction fee
- providerPublicFunds: Funds of public provider
- providerPrivateFunds: Funds of private provider
- providerPrivateClientLimit: Limit of connection to Private provider
- providerPublicMinRate: Minimum rate of public provider
- providerPublicMaxRate: Maximum rate of public provider
- providerPrivateMinRate: Minimum rate of private provider
- providerPrivateMaxRate: Maximum rate of private provider
- providerGasProtect: Limit for scanning Schelling rounds, against gas overflow
- providerInterestMinFunds: Minimal value of Connected Capital at Provider
- providerRentRate: Comission of admin
- schellingRoundBlockDelay: Schelling Rounds block time
- schellingCheckRounds: Retrospective scanning Schelling rounds
- schellingCheckUps: Retrospective checking, how many „UP” needed.
- schellingRate: CORION Token emission rate
- schellingRateOnIco: CORION Token emission rate during ICO
- publisherMinAnnouncementDelay: Announcements minimum block time
- publisherOppositeRate: Vetoes % rate

VIII. CORION TOKEN



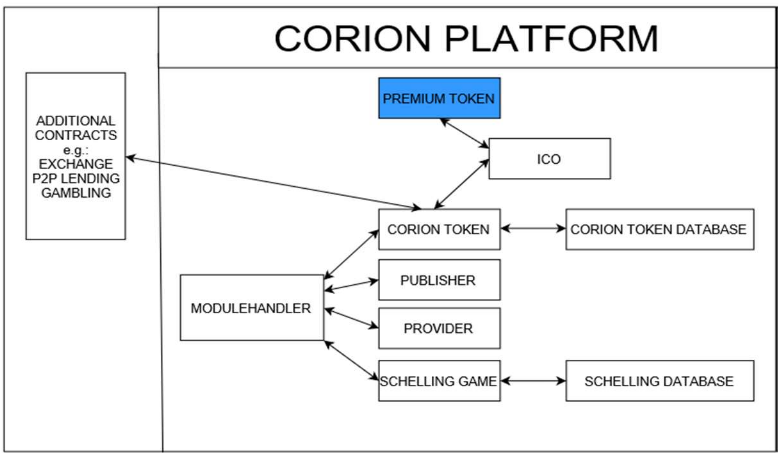
Name: CORION Token
Name short: COR
It can be divided up to six decimal points.
Its database manages every balance of the public keys. The Database is on a separate Contract. This Contract includes the following functions of CORION Token:

- Transaction
- Transaction fee calculation and subtraction
- Allow collection
- Burn
- CORION Token emission

Transaction: CORION Token transfer between two public keys (including collection). Transaction fee will be charged. If the receiver is Contract, than the „Transaction and call” function will be called.

Transaction and call: After transaction the "receiveCORIONToken" function will be called. If the call fails, the transaction will be revoked.
Allow collection: One address may entitle another address to subtract a specific amount of CORIONTokens from his address and credit them on his own. Specific amounts can be partly subtracted. An address may declare more collection addresses.
Allow collection and call: After Allow collection, the Contract is notified that the "approvedCORIONToken" function will be called.
If this call failed, the collection won't be approved.
Transaction fee: 0.02% minimum 0.02 CORION Token and maximum 5 CORION Tokens. 80% of the Transaction fee will be burned, 20% will be transferred to the Schelling game prizepool. It is always the owner's address that pays the transaction fee.
Burn: CORION Token burn
Token emission (mint): Creating new CORION Tokens

IX. PREMIUM TOKEN

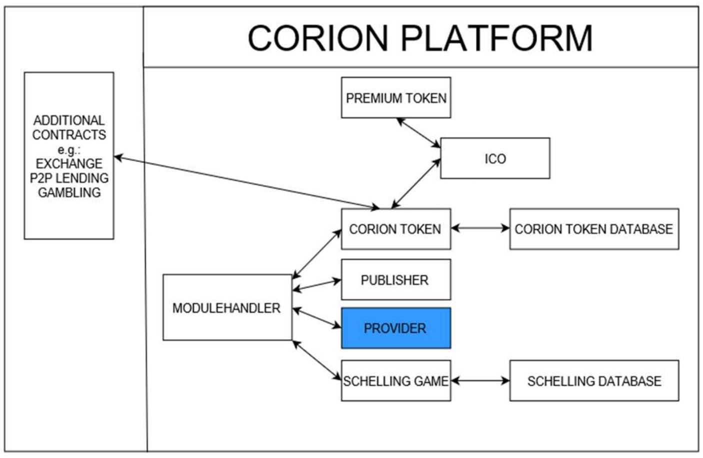


Name: PREMIUM Token
Name short: CORP
It can be defined only as an integer, there is no transaction fee.
Its database manages every balance of the public keys. The Database is on a separate Contract. This Contract includes the following functions of CORION Token:

- Allow collection
- Transaction
- PREMIUM Token issue

Transaction: PREMIUM Token transfer between two public keys (including collection). No transaction fee is charged. If the receiver is Contract, the „Transaction and call” function will be called.
Transaction and call: After transaction the "receiveCORIONPREMIUMToken" function will be called. If the call fails, the transaction will be revoked.
Allow collection: One address may entitle another address to subtract a specific amount of PREMIUM Tokens from his address and credit them on his own. Specific amounts can be partly subtracted. An address may declare more collection addresses.
Allow collection and call: After Allow collection, the Contract is notified that the "approvedCORIONPREMIUMToken" function will be called. If this call fails, the collection won't be approved.
Transaction fee: no transaction fee
Burn: There won't be any PREMIUM Token burn.
PREMIUM Token issue: Only during the ICO period.

X. PROVIDER



Every public key that meets the requirements can be a provider. CORION Token emission will be executed through the Providers. Each public key which is not a provider yet can connect to a Provider and only one Provider at a time to take their share of the CORION Token emission. One public key can become only one Provider.

Connected Capital:
All the individual capitals collected on the addresses and connected to a given Service Provider at the moment of CORION Token emission.

Two types of Provider:
Public
- Any public key can join
- Limitless number of entrants participants
- Emission share rate 30%-70%
- Deposit fund of 3000 CORION Tokens required
- The CORION Token deposit fund on the Provider's public key will not take a share of the CORION Token emission.
- No minimum Connected Capital is required for the CORION Token emission

Private
- Can create a WhiteList with the public keys which are allowed to join
- Only the public keys on the list can join
- Maximum 250 keys
- Emission share rate 0%-100%
- Deposit fund of 8000 CORION Tokens required
- The CORION Token deposit fund on the Provider's public key will take a share of the CORION Token emission
- Minimum 25000 Connected Capital is needed for the CORION Token emission

```
uint256 private minFundsForPublic     = 3000;
uint256 private minFundsForPrivate    = 8000;
uint256 private privateProviderLimit  = 250;
uint8 private publicMinRate           = 30;
uint8 private privateMinRate          = 0;
uint8 private publicMaxRate           = 70;
uint8 private privateMaxRate          = 100;
uint256 private gasProtectMaxRounds   = 630;
uint256 private interestMinFunds      = 25000;
```

The connection is one level deep

Provider	whom others join;
Partner/Client	Who joins one Provider;
Independent	Neither a provider, nor a Partner/Client;

Parameters needed to become a Provider:

priv	Private provider or not
name	Provider’s Name
website	Provider’s URL
country	Provider’s Country
info	Provider’s Introduction
rate	Percentage of the CORION Token emission to be distributed among the users
isForRent	Is the provider for rent or not
admin	Admin address (by renter)

A Contract cannot become a Provider.

Get Reward:

Only the Providers that meet the requirements can take a share of CORION Token emission

getReward(address beneficiary, uint256 limit, address provider)

Parameters:

- beneficiary: Provider can set up a public key to call his CORION Token emission reward on it.
- limit: How many Schelling game rounds must be checked (default 630 rounds)
- provider: Provider’s public key

CORION Token emission depends on Schelling game rounds.

checkReward(address addr)

This function is only for information without fees. It checks if the public key is entitled for CORION Token emission.

Rent a Provider:

Independent public keys can rent a Provider.

In this case isForRent function must be TRUE. The lessor gets 20%, the admin gets 80% of the CORION Token emission.

Public parameters (getProviderDetails and getProviderInfo):

- Name
- URL
- Country
- Info
- Creating date
- Is for rent
- Public key;
- Public or private;
- Emission share rate
- Is active
- Will take a share of CORION Token emission
- Counting participants

Closing a provider and make it inactive:

Provider can be closed only by the Owner, and after the CORION Token emission has been called.

The joined Client must disconnect, after they called their CORION Token emission.

Partner/Client forbidding:
function disallowUsers(address provider, address[] addr)
Private Providers can create a Blacklist, based on the Whitelist. In this Blacklist the addresses which are not allowed to join this Private Provider anymore will be listed.
Partner/Client join to Provider:
Every public key that is not a Provider at the moment:
- Can join a Provider where he is not on the Blacklist
- Can join a private Provider where he is on the White list;
- Can disconnect any Provider and become Independent;

Disconnect from Provider:
Partners/Client can't call their CORION Token emission share only after being disconnected from a Provider.
CORION Token emission through Provider:
New CORION Tokens will be created based on the result of the Schelling game. The new CORION tokens will be distributed among the Partners by the Providers. New Tokens will be automatically generated directly on the Provider's addresses.

Calculating CORION Token emission:
Calculations per Schelling rounds

Calculating Partner/Client share:
*value * globalFunds.reward / globalFunds.supply * rate / 100*

Calculating Provider share based on Connected Capital:
value * globalFunds.reward / globalFunds.supply * (100-rate) / 100

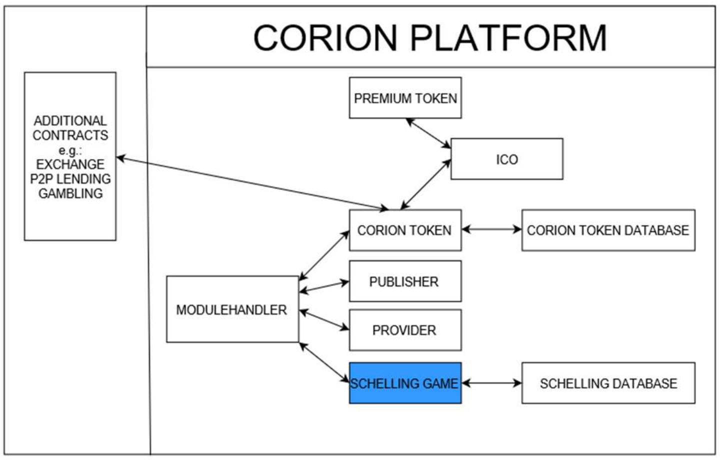
Calculating Provider share based on Deposit fund:
value * globalFunds[a].reward / globalFunds[a].supply

parameters:
- value: The amount of CORION Tokens on the public key at the moment of the CORION Token emission.
- globalFunds.reward: Quantity of emission.
- globalFunds.supply: Connected Capitals of to the emission entitled Providers
- rate: Emission share rate

On the Deposit funds the lesser, on Connected Capital the Admin get the Emission.

Building the Database:
The database of Contract has the data, to calculate, who, when, and how was the balance of the participant in the CORION Token emission.

XI. CORION (Schelling) GAME



This Contract is responsible for CORION Token emission. This is a community based game, and every normal public key (not Contract) being in the CORION Token deposit is allowed to participate. The game is divided by Schelling rounds (772 blocks, ~3 hours) The participants vote on whether the actual rate of CORION Token is ABOVE or BELOW 1 USD. Counting the votes is based on the amount of CORION Token deposit. The votes are encrypted. Votes have to be decrypted in the next ½ Schelling round (386 blocks ~ 1,5 hour), if it does not happen, the vote will be invalidated. After ½ Schelling round the result will be checked. The votes in majority are the winners. The winning votes take their share of the prize pool, the losers' deposits will be burned. If the result is draw, everyone's a winner, and everyone will take their share of the prize pool.

Schelling games steps:
To participate the user has to make a deposit, calling the "addFunds" function.
Steps of the game:
- prepareVote: Sending encrypted vote
- sendVote: Decrypting vote
- checkVote: Counting votes
- getRewards: In case of winning, getting the prize

Schelling game can be played only in this order. New votes can be initialized only after checking earlier votes.
Deposit can be called back with "getFunds" function only after the reward call as well.

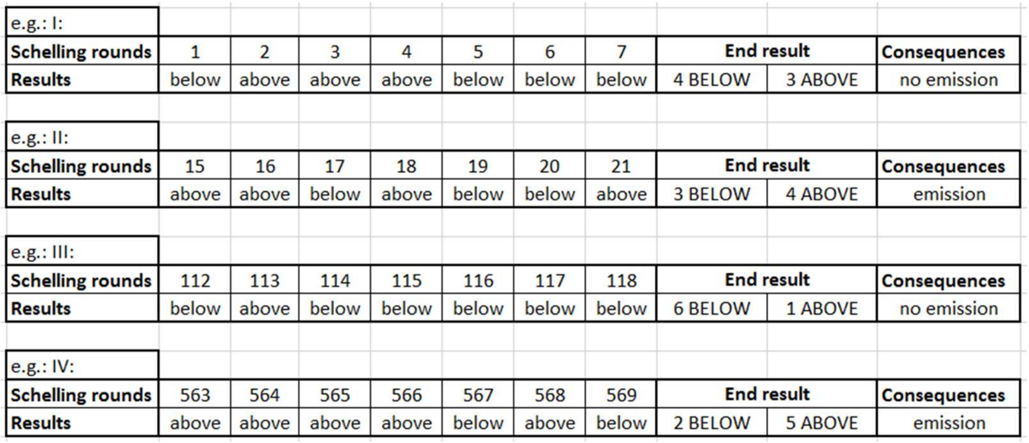
Creating encrypted vote:
First character: 1 or 0 that is, ABOVE or BELOW
Further characters: Random characters
e.g.:
1aebhao43uebcno5uacebbaeuobfae5bnd
It is called a raw vote. It's hash value is:
e63ce750843da86469e3a054a498dd77b8cf4ac91e97cf3323b795039cc4e06d"
This hash will be sent as an encrypted vote.
To decrypt an encrypted vote the raw vote is necessary, because the hash value of both votes will be checked to make sure the values are the same. If not, the voter loses his deposit, if yes, it is counted as a valid vote.

Prize pool:
20% of the transaction fee goes to the prize pool. After checking the vote the whole amount

of the actual prize pool will be shared among the winners. If there is no winner (no votes), the actual prize pool will be added to the next Schelling round prize pool.
Prize pool can be expanded on voluntary basis, when someone decides to transfer CORION Token to Schelling module.
Calculating win share:
Client Deposit * Prize pool / Counts of winning votes
Get rewards:
getRewards(address beneficiary)
Rewards can be called with this function. As a destination address, there can be a beneficiary added. The rewards will be collected till they aren't called.
Check Rewards:
checkReward() returns (uint256 reward, uint256 steps) Function to check the number of rewards. Only for information.

End of Schelling round, CORION Token emission:
The results of Schelling rounds control the CORION Token emission.
The Schelling rounds are persistent.
The main question to be answered: Is the actual CORION Token rate ABOVE or BELOW 1 USD?
The effect on CORION Token emission:
After every CHECKVOTE call, the result of the last 7 Schelling rounds will be checked.
Possible outcomes and consequences:
- Result: 7 BELOW 0 ABOVE – No CORION Token emission
- Result: 6 BELOW 1 ABOVE – No CORION Token emission
- Result: 5 BELOW 2 ABOVE – No CORION Token emission
- Result: 4 BELOW 3 ABOVE – No CORION Token emission
- Result: 3 BELOW 4 ABOVE – CORION Token emission will be initialized
- Result: 2 BELOW 5 ABOVE – CORION Token emission will be initialized
- Result: 1 BELOW 6 ABOVE – CORION Token emission will be initialized
- Result: 0 BELOW 7 ABOVE – CORION Token emission will be initialized

e.g.:



e.g.: I:										
Schelling rounds	1	2	3	4	5	6	7	**End result**		**Consequences**
Results	below	above	above	above	below	below	below	4 BELOW	3 ABOVE	no emission

e.g.: II:										
Schelling rounds	15	16	17	18	19	20	21	**End result**		**Consequences**
Results	above	above	below	above	below	below	above	3 BELOW	4 ABOVE	emission

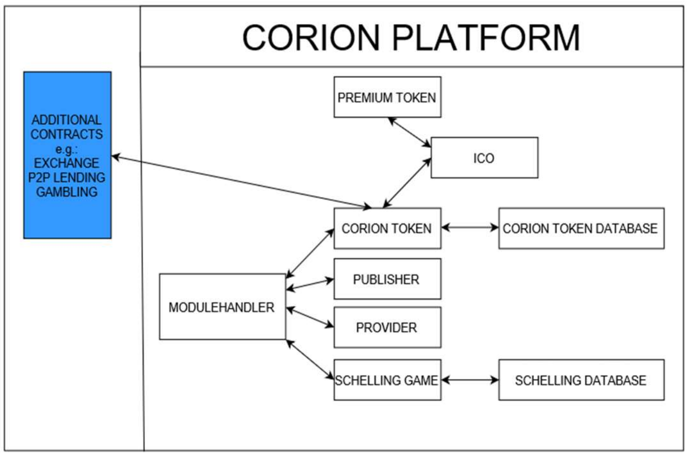
e.g.: III:										
Schelling rounds	112	113	114	115	116	117	118	**End result**		**Consequences**
Results	below	above	below	below	below	below	below	6 BELOW	1 ABOVE	no emission

e.g.: IV:										
Schelling rounds	563	564	565	566	567	568	569	**End result**		**Consequences**
Results	above	above	above	above	below	above	below	2 BELOW	5 ABOVE	emission

Amount of CORION Token emission:
The emission will be executed in the Provider Contract. Its value is 0.3% of the whole CORION Token quantity.
e.g.:
- Total CORION Token quantity (TotalSupply) = 100,000,000 COR
- CORION Token emission rate = 0.3%

- Quantity of CORION Tokens will be newly generated = 100,000,000* 0.3% = 300,000 COR
- This 300,000 COR will be distributed among those entitled (Providers and their connected users)
- Quantity of CORION Tokens after emission = 100,300,000 COR

XII. BUILDING SERVICES ONTO CORION TOKEN / PLATFORM



Anyone can use the functions of CORION Platform with Ethereum Classic SMART Contracts. Not only the normal addresses, but those SMART Contracts can contain Tokens as well.

The following function will be called by 3rd party SMART Contracts, if it contains CORION Token or collection permission:
function receiveCORIONToken (address, uint256, bytes) external returns (bool, uint256) {}
Parameters:

- address	Sender's address
- uint256	CORION Tokens amount
- bytes	Data defined by the sender
- bool	Result of the call. It will be defined by the 3rd party SMART Contract. If it is FALSE, the transaction fails.
- uint256	Token-back quantity. If 3rd party Contract didn't use the total CORION Token amount, there is an option to send the rest back.

function approvedCORIONToken(address, uint256, bytes) external returns (bool) {}
Parameters:

- address	Sender's address
- uint256	CORION Tokens amount
- bytes	Data defined by the sender
- bool	It will be defined by the 3rd party SMART Contract. If it is FALSE, the transaction fails.

If 3rd party SMART Contract contains CORION Tokens, it can use the features of CORION Platform, except for the Schelling game.
Programming SMART Contract onto CORION Platform is at your own risk only.
CORION Platform won't take any responsibility for 3rd party SMART Contracts.

CORION Token use ERC223 standard. Further information:
https://github.com/Ethereum/EIPs/issues/223
https://github.com/Ethereum/EIPs/issues/20

XIII. EXCHANGE

CORION Exchange functions as a 3rd party SMART Contract.
On the CORION Exchange, CORION Tokens can be sold and bought for ETC with P2P method, without any other participant.
A CORION Exchange Contract operates like a 3rd party SMART Contract. Any CORION Exchange function will be verified by Ethereum Classic, any fraud or theft is impossible.
All the rights to change, modify or close the CORION Exchange Contract are reserved for the Owner.
CORION Platform use this CORION Exchange Contract to buy gas.

There are options to sell or buy CORION Tokens instantly or at a fix rate.

Settings:

- fee: Exchange fee.
- feeM: Coefficient of the fee is required because of the Solidity language.
- orderUnit: The least trade unit. Any value is rounded off to the value of this unit. It is the number of CORION Tokens.
- rateStep: The least rate difference unit. Any value is rounded off to this value. It is the number of Ethers.
- maxForGasSell: Max amount of available Gas
- sellPrice: The lowest sell price at the moment.
- buyPrice: The highest buy price at the moment.
- disabled: State of Contract, if it is TRUE, position can only close, and transfer the balance.

Filling up the Balance:
Each address has their own, separate Balance in the database (CORION Token and Ether).
CORION Tokens and Ether balance will be transferred to the Exchange address to fill it up.
When a CORION Token transfer is executed a transaction fee will be charged.

Balance withdrawal:
function payout(uint256 amount, bool eth)
Parameters:

@amount	Amount of withdrawal
@eth	If its TRUE → Withdraw from Ethereum Classic address, if it's FALSE → Withdraw from CORION Token address.

In case a CORION Token transfer is executed, a transaction fee will be charged.

CORION Token sell:
function sell(bool token, uint256 value, bool instant, uint256 rate)
Corion Token sell instantly or at a fix rate.

CORION Token buy:
function buy(bool token, uint256 value, bool instant, uint256 rate)
Corion Token buy instantly or at a fix rate.
Parameters:

@token	Token switch. TRUE = Token; FALSE = Ether
@value	Amount for trade. Can't be more than the actual Balance.

@instant	Is the offer instant or not
@rate	Rate. Mandatory if not instant trade

During the trades, the following rounding rules are in effect:
- „orderUnit"
- „rateStep"

Rules of a delayed (not instant) trade in the aspect of Rates:
- Sell will be executed at the given rate or above it
- Buy will be execute at the given rate or under it

Rules of a delayed (not instant) trade in the aspect of Quantity:
 - If only a part of the offered Quantity can be traded, a new position will be opened for the rest of that Quantity

<u>**e.g.:**</u>
- Want to sell 100 COR at 1.1 minimum Rate
- Successfully sold 70 COR from 100 COR →30 COR remain
- New position for the rest of 30 COR at 1.1 minimum Rate

Rules of instant trade in the aspect of Quantity:
- If only a part of the offered Quantity can be sold or bought, there won't be a new position opened.

<u>**e.g.:**</u>
- Want to sell 100 COR instantly
- Successfully sold 70 COR from 100 COR -> 30 COR remain
- If the rest of 30 COR are to be sold as well, a new instant sell position has to be opened

Instant trade switch:
- TRUE:
    - Sell: The transaction will be at the highest selling rate realized
    - Buy: The transaction will be at the lowest buying rate realized
- FALSE

    Token sell switch:
    - TRUE: The amount of offered CORION Tokens has to be in CORION Token given
    - FALSE: The amount of offered CORION Tokens has to be in Ether given

    Token buy switch:
    - TRUE: The amount of offered CORION Tokens has to be in CORION Token given
    - FALSE: The amount of offered CORION Tokens has to be in Ether given

There is an exchange fee charged per transaction on both sides of the trade.

Modify position:
function moveOrder(uint256 orderID, uint256 newRate)
Current position will be closed, and a new one will be opened with a new rate.

Cancel position:
function cancelOrder(uint256 orderID)

Simulation:
function getSell(bool token, uint256 value)

function getBuy(bool token, uint256 value)
Ratesimulation. Free of charge function.

Calling position parameters:

Based on the position ID, the parameters of the positions can be checked.
function getOrder(uint256 orderID) returns (address owner, uint256 amount, uint256 rate, bool sell)
Parameters:

@orderID	Positions ID
@owner	Owner
@amount	CORION tokens amount
@rate	Rate
@sell	Sell or buy

----v0.4----020517----