

# Coreum: One Blockchain For All

## Technical White Paper

Version 1.01 - June 2022

**Disclaimer** Please read the entirety of this "Disclaimer" section carefully. Nothing herein constitutes legal, financial, business or tax advice and you should consult your own legal, financial, tax or other professional advisor(s) before engaging in any development and activity in connection herewith. Neither Sologenic development Foundation Limited, any of the project team members (the **CORE** team) who have worked on the Coreum blockchain (as defined herein) or project to develop the Coreum blockchain in any way whatsoever, any distributor/vendor of \$coretokens, including without limitation Sologenic Development Foundation Limited, nor any service provider shall be liable for any kind of direct or indirect damage or loss whatsoever which you may suffer in connection with accessing this whitepaper, the website at <https://coreum.com/> (the **Website**) or any other websites or materials published by the foundation and the team. This whitepaper is subject to change without any prior notice.

### Abstract

The advent of blockchain technology has attracted huge interest in modern times. Through various protocols, the blockchain plays an important role in every business vertical. Some blockchains such as Bitcoin are a great store of value, others like Ethereum promise to verify and execute application code. While smart contract engines such as EVM are great for numerous types of applications, they remain non-deterministic and non-scalable.

The Coreum blockchain is distinct in many ways and incentivizes the network participants to conduct more transactions by providing bulk fee discounts. It is designed to solve real-world problems at scale by providing native token management systems, such as a Decentralized Exchange (DEX), while being fully decentralized. In addition to the built-on-chain solutions, Coreum uses WebAssembly (WASM) to process smart contracts, and utilizes the Tendermint Byzantine Fault Tolerance (BFT) consensus mechanism and Cosmos SDK's proven Bonded Proof of Stake (BPoS).

Furthermore, Coreum is built for token ecosystems such as digital assets issuing, stablecoins, traditional asset tokenizations, CBDCs, and NFTs.

## Table of content

Disclaimer	1
Abstract	1
1 Architecture	4
1.1 Tendermint & Cosmos SDK	4
1.2 Interoperability with Other Blockchains	5
1.3 Side Chains	6
2 Proof of Stake	6
2.1 Staking	7
2.2 Validators	8
2.3 Delegating	9
2.4 Transaction Fees	10
2.4.1 Gas	10
2.4.2 Gas Price	10
2.5 Validator Rewards	12
2.5.1 Reward Distribution	12
2.6 Slashing	13
3 Governance	14
4 Token Management	14
4.1 Introduction	14
4.2 Architecture	15
4.3 Token Properties	15
4.3.1 Token Symbol	15
4.3.2 Minting & Burning	15
4.3.3 Whitelisting	15
4.3.4 Fungible vs Non Fungible	16
4.3.5 Token Freezing	16
4.3.6 Precision	16

4.4	Token Issuance	16
4.4.1	Transactions	17
4.4.2	Complete Flow Example	17
5	Smart Contracts	18
5.1	WASM	19
5.2	CosmWasm	19
5.3	CosmWasm Architecture	19
5.4	Contract Flow	20
5.5	Cosmos SDK Wasm Module	20
5.6	Rust Language	21
6	Decentralized Exchange (DEX)	21
6.1	On-Demand Orderbook	21
6.2	Direct & Indirect Order Matching	22
6.3	Order Execution	22
6.4	Order Types	23
6.4.1	Market Orders	23
6.4.2	Limit Orders	23
6.5	Advanced Features	23
6.5.1	Good Till Cancel Orders	23
6.5.2	Good Till Time Orders	23
6.5.3	Immediate or Cancel Orders	23
6.5.4	Fill or Kill Orders	24
7	Decentralized Apps (dApps)	24
8	Use Cases	24
9	Token Economy	25
10	Allocation	25
11	References	26

# 1 Architecture

At the heart of every blockchain lies the consensus mechanism; Coreum utilizes Tendermint - a well-established consensus engine that many blockchains rely upon. The Cosmos SDK makes use of Tendermint to provide tooling and pre-made modules for the building of application-specific Blockchains with a proof of stake security mechanism.

Coreum will be built on top of Tendermint and Cosmos SDK while making changes whenever necessary. This will allow the new layer-1 to tap into the vast ecosystem built around the Cosmos SDK; such as wallets, explorers, and other modules.

## 1.1 Tendermint & Cosmos SDK

*Excerpted directly from tendermint website:*

“Tendermint is software for securely and consistently replicating an application on many machines. By securely, we mean that Tendermint works even if up to  $\frac{1}{3}$  of machines fail in arbitrary ways. By consistently, we mean that every non-faulty machine sees the same transaction log and computes the same state. Secure and consistent replication is a fundamental problem in distributed systems; it plays a critical role in the fault tolerance of a broad range of applications, from currencies, to elections, to infrastructure orchestration, and beyond.” [1]

Although in the above paragraph from tendermint website it is said that Tendermint works when even up to  $\frac{1}{3}$  of machines fail, but in reality tendermint consensus works with voting power and not number of machines, and it is up to the application layer to decide how that voting power is determined. One might assign 1 voting power to each machine on which the above statement is based or in the case of Coreum voting power is assigned proportionally to the amounts of CORE staked.

“More formally, Tendermint Core performs Byzantine Fault Tolerant (BFT) State Machine Replication (SMR) for arbitrary deterministic, finite state machines.” [2]

Tendermint brings decades-old academic research on BFT into the world of blockchain and facilitates high-throughput low-latency proof-of-stake blockchain, which is not only fast but also hugely more power efficient than proof-of-work.

Tendermint has two main parts: a consensus engine and an interface to communicate with the application layer. The application layer will contain all the business logic specific to the blockchain and the consensus layer will facilitate networking and consensus between all nodes. And this is where Cosmos SDK fits in. It facilitates building applications on top of tendermint by providing an architectural pattern, tooling and other commonly used modules.

## 1.2 Interoperability with Other Blockchains

Coreum will enable communication to, and, from other blockchains. Users will be able to transfer tokens from external blockchains into Coreum and vice versa. This interoperability works by locking or burning the tokens in the source chain and minting them on the target chain. Thus, when the token is transferred back from the target chain onto the source chain, it is burned on the target and unlocked in the source.

For Cosmos-SDK-based chains, Coreum will use the IBC protocol [3] which is available as an open source component. IBC is an interoperability protocol designed for communicating arbitrary data between arbitrary state machines. It consists of 2 layers:

1. Data transfer layer: establishes communication channels by creating connections and transferring data between chains.
2. Application layer: which defines how messages should be encoded, decoded and interpreted at each end of the communication channel.

The IBC protocol can be used for different use cases including but not limited to transfers, interchain accounts (delegate calls between two chains) and oracle data feeds.

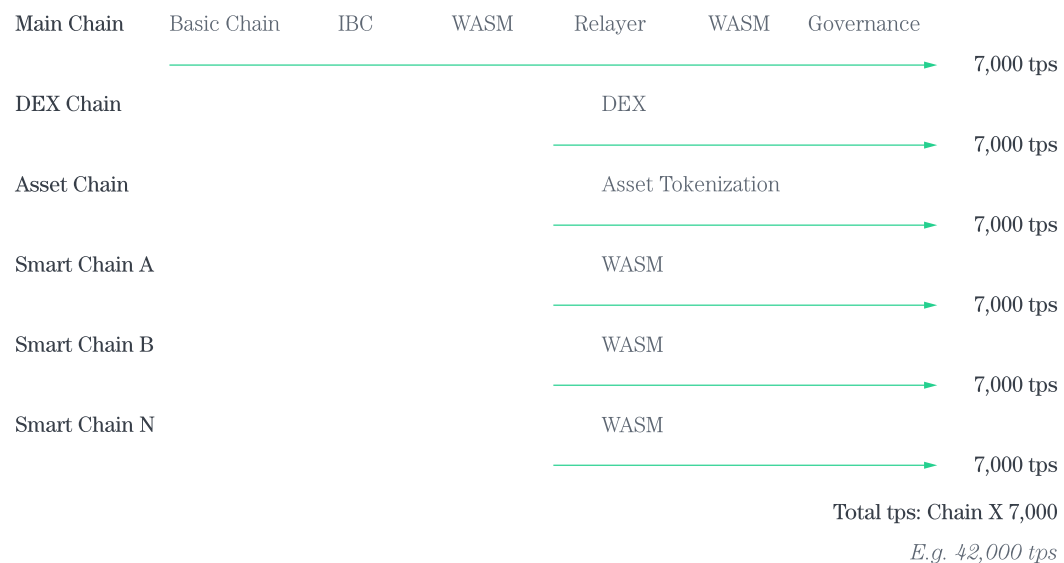
Coreum will implement communication layers either via IBC or bridge to communicate with the following blockchains with more to follow down the line:

- Bitcoin
- Ethereum
- Ripple
- Solana
- Binance Chain & Binance Smart Chain
- Cosmos (ATOM)

### 1.3 Side Chains

In Computer science there is an invariable limit on how much you can scale a software by adding more resources to it called vertical scaling. The solution to that problem is a parallelism formally known as horizontal scaling.

Coreum is incurring heavy research and development around the use of side chains for exponential scalability and the production of more throughput when needed. A single chain can do up to 7000 simple transactions per second, consequently, by adding a single side chain, the throughput is increased to 14000. Of course there are applications that cannot run on multiple side chains, for example a DEX. For such use cases, a dedicated chain will run individually. The image below provides an overview of how side chains will look like.



## 2 Proof of Stake

As mentioned before, Coreum will use Cosmos SDK's Bonded Proof of Stake security scheme to build its own blockchain. A short description of the most important parts of the BPOS will be provided here.

## 2.1 Staking

Staking is one of the most important components of a Proof-of-Stake Blockchain. It ensures the network remains secure while giving validators and delegators a passive income by staking their CORE tokens and getting rewards.

In fact, Staking is the backbone of every proof-of-stake consensus. A proof-of-stake network is secured when a stakeholder locks their tokens and gets voting power in exchange, then they use that power to vote on each block. The assumption is that it is in the benefit of the stakeholders to secure the network since the value of their assets depends on the security of the network itself.

It can be deduced that the more tokens are locked as stake, the more secure the network is. For example if the total value locked (TVL) as stake is about 1% of the entire token value, then if an attacker controls one third of that 1%, they can halt the network, but if 75% is locked as TVL then they need to control 25% of the total value to halt the network. So a proof of stake network must encourage its participants to stake and reach a reasonably high TVL percentage.

Given the fact that the only incentivization mechanism in proof-of-stake is block rewards, then if there are enough transactions in each block, more fees will be collected and more people will stake their tokens. But if there are not enough transactions happening then the rewards will be too low and the TVL will go down, bringing risk to the network. To accommodate for that problem, variable inflation is introduced. Meaning that some tokens will be minted in each block, added to block rewards (alongside fees), and passed to the validators and delegators to incentivize them to provide more stake, and the amount of inflation rate is inversely proportional to TVL. If TVL is equal or higher than the target value then the inflation rate goes down to its minimum and if it is not will go up but there is an upper limit to the inflation rate.

These tokens will be minted in a manner that a maximum yearly inflation rate is observed. It is evident that the minting will introduce inflation to the system, which will further incentivize non-stakers to effectively stake their tokens so their assets don't lose value. It must be noted that if there are enough transactions in each block, and, enough incentivization is produced from transaction fees so that stakers are locking enough stake to meet the target TVL, then the inflation will not be necessary and will automatically go down to zero.

## 2.2 Validators

In Tendermint, validators are responsible for the seamless operation of the network by ensuring that blockchain invariants are enforced and consensus is reached. They do so by putting their assets as stake, which grants them voting power proportional to the amount of their stake. Delegation is another source of getting voting power which will be discussed later.

The exact procedure of Tendermint consensus is described in the Tendermint whitepaper [1] & their website [2].

In Coreum there are different types of validators as:

- **Public validator:** Introduced to allow any network user to participate in the consensus and contribute to the security of the network. To increase decentralization, any network participant can become a public validator by staking a minimum amount of CORE tokens. Individuals can also step down from being a public validator; however, their stake will stay locked for some time after that called unbounding period.
- **Super validator:** Any public validator can become a super validator by submitting a proposal via on-chain governance and getting more than 50% of votes from all validators. The reasoning behind introducing Super Validators is to add more trust to the circle of validators by introducing social trust. In other words, having a mix of Super and Public Validators will allow Coreum to have the same level of security with less validators compared to only having public validators which, in turn, will translate in delivering more throughput and less latency while keeping the same security guarantees. A super validator at any time may decide to step down and go back to being a public validator. And it is also possible for other token holders to submit a proposal to demote other super validators from being a super validator.
- **Active validator:** A subset of validators will be chosen for a period of time called Active Validator Interval (AVI) to partake in the validation process. The active validators will be chosen from public and super validators with a fixed number of seats for each validator type. The active validators will rotate to give every validator a chance to participate in the process and earn rewards, which will increase decentralization and network participation. The motivation behind the Active Validator concept is to limit the number of validators that are actively participating on the consensus so Coreum can deliver high throughput at all times.



There will be 16 active validators on the Coreum blockchain including 9 public and 7 super validators. With a fixed number of active validators, and an arbitrary number of public and super validators, a round robin mechanism will be used to choose the active ones from the pool of public and super validators for the duration of the Active Validator Interval (AVI).

This means that there will be two queues, one for each validator set (public and super validator), each of which contains a pair of validators together with their respective priority number. The steps listed below will be carried out in each queue:

- At the genesis block, preconfigured validators will have some voting power allowing blockchain to start.
- At each block, token holders may propose one of the following actions:
  - Become public validator
  - Step down from being public validator
  - Submit a proposal to become a super validator. The proposal will be put into vote via governance & accepted if the majority of votes are in favour
  - Step down from being a super validator
  - Submit Proposal to demote other super validators from their position. The proposal will be put into vote via governance and get accepted only if enough votes are acquired.
- At the end of each AVI, in each queue, an algorithm will run to determine the next selection of active validators. A basic description of the algorithm follows, but the complete implementation is more nuanced.
  - Top validators with the highest priority number will be chosen as the next validator and their priority will be decreased by the total voting power present in the queue.
  - Every other validator in the queue will have their priority increased equal to their voting power.

In order for an entity to become a validator, it must run a full node and have a stake of at least 10,000 CORE tokens.

## 2.3 Delegating

Users who do not wish or do not have the means to become a validator can become delegators. In short, delegators can choose a validator and stake their CORE

tokens by paying a commission to the validator. Delegators will also get punished if the validator misbehaves, so they must carefully choose who they delegate to, to avoid punishments.

Delegators can also choose to delegate to multiple validators by consigning a portion of their stake to each one. This will help reduce the financial risk of getting slashed if one of the validators misbehaves so not all of their assets would be subject to a single slashing.

Moreover, delegators are also expected to actively participate in governance. A delegator's voting power is proportional to the size of their stake with the validator and if they don't engage in community voting, their power is shifted to the validators.

## 2.4 Transaction Fees

Coreum uses fees for transaction processing to secure the network by paying validators, disincentivizing attacks, etc. The fee is determined by the following formula:

$$fee = gasPrice * gasUsed$$

The `gasUsed` variable is invariant for a known transaction but `gasPrice` may change which will result in having different fees at different times for that transaction.

### 2.4.1 Gas

Gas is a measure of how much computational power each transaction needs. For each transaction type, the amount of gas needed is a predetermined amount. The total gas allowed in each block will be called `MaxBlockGas`.

### 2.4.2 Gas Price

Coreum will use a combination of governance and an on-chain mechanism to dynamically determine the `gasPrice` to be specified in units of CORE. Some variables are defined below to utilize in gas price calculations:

**Base Gas Price ( $GP_0$ ):** gas price when the average block load is zero, determined by on-chain governance.

**BlockLoad:** Is an indicator of what percentage of block capacity is used and is determined by the following formula:

$$BlockLoad = GasConsumedInBlock / MaxBlockGas$$

**AverageBlockLoad<sub>n</sub>:** Is the average BlockLoad in the previous n blocks

**BoundaryLoad:** Is the ideal BlockLoad that we want the blockchain to operate at.

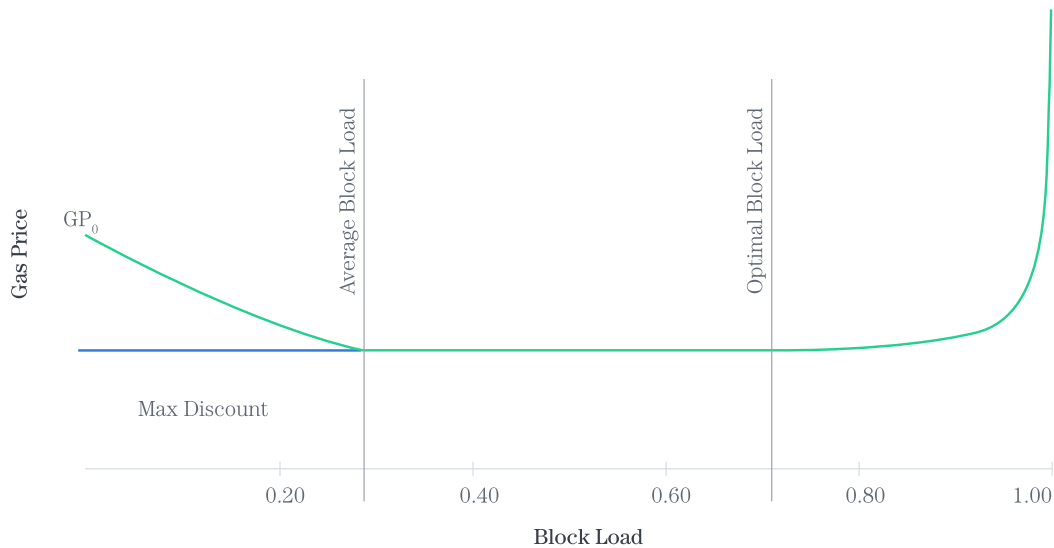
**MaxDiscount:** The maximum discount applied on Base Gas Price when the network is operating below the BoundaryLoad.

$$gasPrice = \begin{cases} GP_0 * (1 - maxDiscount)^{BlockLoad/BoundaryBlockLoad}, & \text{if } BlockLoad < AverageBlockLoad_n \\ GP_0 * (1 - maxDiscount), & \text{if } AverageBlockLoad_n \geq BlockLoad \geq BoundaryBlockLoad \\ GP_0 * (1 - maxDiscount) * \frac{1 - BoundaryBlockLoad}{1 - BlockLoad}, & \text{otherwise} \end{cases}$$

The figure below demonstrates the above equation. In each block, BlockLoad is used to determine the gas price of the next block based on the Base Gas Price which in turn is determined by on chain governance. The equation is piecewise, with the first piece corresponding to a section with a downward trend. Here the *BlockLoad* ranges from zero to *AverageBlockLoad<sub>n</sub>*, and is designed to incentivize the blockchain users to collectively do a little more (or at least not less) transactions than the average block, and benefit from the maximum discount of gas price provided by the network.

The second part of the equation is a constant value equal to maximum discount applied to the base price. It corresponds to the flat line in the middle of the chart. This is the section that the users are encouraged to keep the network running at.

The third section is where the *BlockLoad* is higher than the *BoundaryBlockLoad* and the network will increase gas prices to disincentivize more transactions getting submitted and bottlenecking the network. Another word for this section of the equation is *fee escalation* and it is designed to keep the system from overloads.



## 2.5 Validator Rewards

Block rewards will be split between active validators to incentivize their participation in the consensus. These rewards consist of transaction fees that go into the block plus additional rewards minted by the blockchain itself. Given the fact that a POS system is secured via its own stake, the minted amount is meant to encourage token holders to stake their tokens in case that there are not enough CORE tokens locked in as stake. So the minted value will go up if TVL is less than the target TVL (up to a maximum yearly inflation) and will go down if TVL is higher than target TVL (and it may even go down to 0).

### 2.5.1 Reward Distribution

Rewards first will be split between public validators and super validators according to their seats. Meaning if there are 9 public validators and 7 super validators then 7/16 of the rewards will go to super validators and 9/16 to public validators. It will then be split among each set in proportion to their stake. For example if all the super validators have 10,000 stake and one of them has 1000, then that validator will get a portion of block rewards equal to  $1/10 * 7/16$ .

## 2.6 Slashing

As important as it is to incentivize the contribution of validators by rewarding them, it is important to punish them on certain occasions for bad behavior; this protocol is called slashing. The mechanism for slashing is taken directly from the Cosmos SDK module with the same name and readers can refer to that for more details. But a general description will be provided here as well.

Punishments related to slashing include disqualifying a validator from the committee board, capturing or burning some of the validator stakes or disabling a validator for a brief period of time.

There are two main types of penalties when it comes to slashing:

1. **Jailing:** The misbehaving validator is immediately put into “jail”, meaning that they cannot participate in next blocks voting until some time has passed and they unjail themselves by issuing a specific transaction.
2. **Slashing:** a percentage of the staked assets of the validator (alongside the delegators who delegated to that validator) will be deducted and burned.

There are two main types of misbehavior:

1. Liveness misbehavior (i.e not showing up to vote on a block)
2. Counterfactual signing misbehavior (e.g double signing, rejecting a valid block,...)

The liveness misbehavior is taken to be unintentional but the counterfactual signing is assumed to be a malicious one.

In either case when a misbehavior is detected the validator immediately will be put into jail. Meaning that they will not be able to participate in future validation processes until some time has passed and they unjail themselves. This ensures that the network is taking immediate action and there is always a healthy set of validators securing the network.

The slashing part works differently for malicious behavior (e.g counterfactual signing) and unintentional misbehavior (e.g liveness). For unintentional misbehavior the validator will be put into jail and slashed immediately and they can unjail themselves after a short period of time. But for the malicious behavior they are kept in jail for a longer period of time and the network will wait until all the evidence for

the misbehavior is gathered and submitted to the network. Slashing will not stack, and only the single most severe penalty will be applied to reduce the harshness of the punishment.

## 3 Governance

Coreum's Proof of Stake Consensus Mechanism (BPoS) allows for an on-chain governance ecosystem that enables stakeholders to vote on various decisions to upgrade and improve the chain over time. The Coreum community can decide on protocol changes and vote on new proposals to the chain. As some blockchains like Ethereum do not offer any sort of on-chain governance features, Coreum engages the community to participate in key decisions.

A range of different proposals can be submitted through on-chain governance and be put up for voting, the list includes but is not limited to: The base gas price, slashing, staking, delegation, number of validators, block times, grants, etc.

The following is a life cycle of proposals on the Coreum blockchain:

1. **Proposal submission:** Stakeholders submit proposals with a fee. Once a proposal reaches a certain deposit, the proposal enters into a voting period.
2. **Voting:** Participants can vote on proposals that reached minimum fee requirements and are active for voting.
3. **Inheritance and penalties:** Delegators inherit their validator's vote if they don't vote themselves.
4. **Claiming deposit:** Users that deposited on proposals can recover their deposits if the proposal was accepted OR if the proposal never entered the voting period.

To achieve this, Coreum will inherit Cosmos SDK's gov module [\[5\]](#) and will make modifications if necessary.

## 4 Token Management

### 4.1 Introduction

The Coreum token has a rich set of properties, which allow users of the blockchain to accomplish a wide range of use cases. For example, trading stock shares, pay dividends on those shares or tokenize a wide range of fungible and non fungible assets.

## 4.2 Architecture

Creating and minting tokens (fungible and non-fungible) on the Coreum blockchain is supported natively. The built-in functionalities allow token issuers to customize their tokenized assets with optional features such as wallet whitelisting, burning and freezing for when it comes to the heavily regulated financial markets like Stocks and ETFs.

## 4.3 Token Properties

### 4.3.1 Token Symbol

Each token in Coreum is uniquely identifiable by the combination of its currency code and issuer. The code derives from taking the address of the issuer, decode it from bech32 to bytes and then encode again using the token symbol as bech32 prefix.

The result would be:

*tokenabcf360al7er8flap94qnl7xghq40d8zez9xlef3j*

### 4.3.2 Minting & Burning

Minting is the process of creating new tokens. Upon issuing a new token, the user will need to indicate how many will be minted and if further minting will be allowed.

On the other hand, tokens are removed from circulation by being “burned”. Users can control if tokens can or cannot be burned.

### 4.3.3 Whitelisting

Whitelisting restricts the usage of a given token to a subset of authorized addresses. This feature enables some token creators to comply with regulations that

require KYC/AML and can be extremely useful for institutional and governmental user types.

At the time of token issuance, the user can specify whether holding (sending/receiving) and trading of this token is allowed by anyone or should be restricted to users who are whitelisted.

#### 4.3.4 Fungible vs Non Fungible

Upon issuance, users will choose whether the token will be **fungible or not**. Non fungible tokens (NFTs) can represent ownership over digital or physical assets and have a maximum supply of 1 token.

Some examples are the following:

- Intellectual properties — unique monkey faces, avatars
- Physical property — land, houses, artwork
- Financial assets — loans, burdens and other responsibilities

#### 4.3.5 Token Freezing

An account freeze essentially means suspending the ability to transfer and receive account tokens. Token issuers can define if an asset can be frozen by its issuer on a specific account. Moreover, the freeze can be applied to specific addresses or to all token holders (global freeze), and it can be applied to the total balance of an address or only to a portion of the balance.

#### 4.3.6 Precision

Upon issuing new tokens, users should indicate the number of decimals a given token can handle.

### 4.4 Token Issuance

The term token issuance may also refer to the process of tokenization, in which an asset outside of the cryptocurrency ecosystem is added to the blockchain via a specific crypto token. In such cases, token issuance becomes the process of creating a token, yet not one that belongs to a cryptocurrency, but rather a token that represents an outside asset. This can have numerous applications on different



asset types (physical, digital or financial).asset types (physical, digital or financial).

Token issuance is the process of creating new fungible and non-fungible tokens on the Coreum blockchain.

With a whitelisting feature, Coreum can offer the means to allow users to comply with regulated environments; some potential users for this feature would be large organizations or governmental entities.

#### 4.4.1 Transactions

- **Issue transaction:** Tokens are issued when minted for the first time on the blockchain. The token code needs to be unique and the token configuration gets saved on a key value store which will be kept on-chain.
- **Mint transaction:** The mint transaction will add a specified amount of tokens to the total supply. The newly minted tokens will go to the specified address. (\*)
- **Burn transaction:** The burn transaction will remove existing tokens from total supply, subtracting them from the supplier's account. (\*)
- **Whitelist transaction:** The whitelist transaction will allow certain addresses to hold (send / receive) and trade a token in which whitelist the address is included. It is important to note that addresses can represent either a blockchain user or smart contract. (\*)

(\*) The mint, burn and whitelist transactions may be available or not depending on options set during Issue transaction.

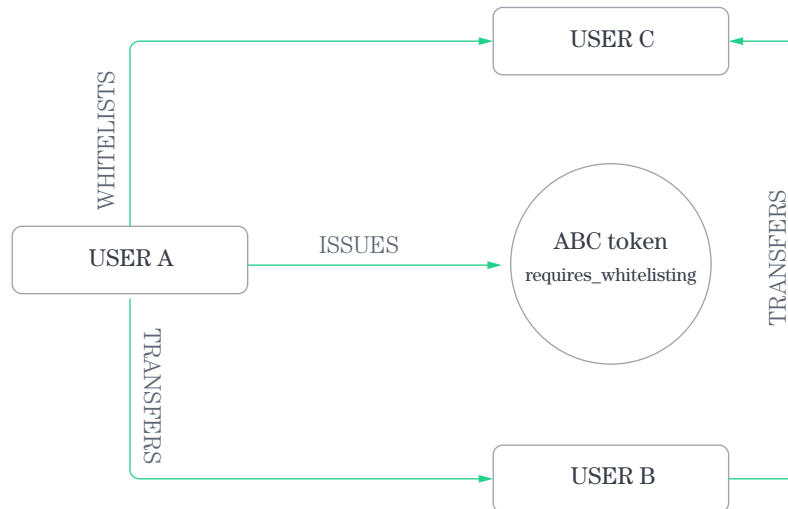
#### 4.4.2 Complete Flow Example

User A issues a new token called “ABC” and sets the configuration in a way that the newly minted tokens are transferred to User B. User A also restricts holding of this token by setting a config setting called “require\_whitelisting”.

After this transaction, User B instantly gets whitelisted and is delivered the “ABC” tokens. Now user C wants to receive “ABC” tokens from user B. They now must ask user A (creator of the token) to whitelist User C to be able to hold the token. After whitelisting is done, user C can receive tokens from user B. This feature enables some token creators to comply with regulations that require KYC/AML.

The creator of the token (user A) can also freeze anyone’s “ABC” tokens held in

their wallets (given that they set the configuration that allows them to do so when they minted the token). The same user (user A) can also set a global freeze to all holders of the “ABC” token at any time (again, given that they set this config when minting the token).



## 5 Smart Contracts

The Coreum blockchain will be able to store and execute smart contracts. In essence, a smart contract is a computer program that can be stored in a blockchain with the purpose of later being instantiated and executed. Although literature on smart contracts dates back to 1994, when Nick Szabo [6] first introduced the concept, Ethereum was the pioneer in making a widely adopted implementation.

While EVM (Ethereum Virtual Machine) was great at the time for smart contract execution, it has shown some shortcomings in various aspects. Some of these shortcomings are:

Some of these shortcomings are:

- Security flaws (re-entrancy attacks, Arithmetic Over/Under Flows, etc). [7]
- Lack of support for integers smaller than 256 bits, which leads to inefficiencies.
- Tight coupling between contract and Solidity language.

## 5.1 WASM

We believe WebAssembly is a much greater engine for smart contract execution and has great scalability and support. It was developed by the W3C (World Wide Web Consortium) with support from companies such as Mozilla, Google and others. While being portable, Turing complete and efficient, developers can write their smart contracts in many different programming languages such as C/C++, C#, Rust, Javascript, Typescript, Haxe, Kotlin and Go.

WASM is fast, efficient, open, debuggable, platform-independent and memory-safe which makes it perfect for smart contract execution.

## 5.2 CosmWasm

Since Coreum blockchain relies heavily on the Cosmos ecosystem, CosmWasm is the chosen platform to handle smart contracts. CosmWasm is a smart contract engine and an important part of Cosmos infrastructure. It is written as a module that can plug into the Cosmos SDK. It facilitates the execution of smart contracts in different chains using the Cosmos's Inter-blockchain communication protocol.

CosmWasm is designed to be a multi-chain solution for building smart contracts that allow the execution of the same contract in different chains. Just by writing a CosmWasm contract you can run contracts on the whole Cosmos ecosystem.

## 5.3 CosmWasm Architecture

CosmWasm fully embraces the actor model [8], in which messages behave in a fire-and-forget manner, they do not wait for any promises to be fulfilled, removing the danger of race conditions. Actors send messages through a message dispatcher as a communication mechanism.

The implementation of Actor model pattern adds the following added value:

- Increased security: Since it prevents contracts from calling each other it avoids reentrancy attacks. Contracts are allowed to message each other, but not to be called directly.

- Inter-blockchain messaging: Sending cross chain messages through the IBC is achievable.
- Ease of serialization: Messages can be serializable to many different formats so they can be integrated with external systems.

## 5.4 Contract Flow

The contract lifecycle has three phases:

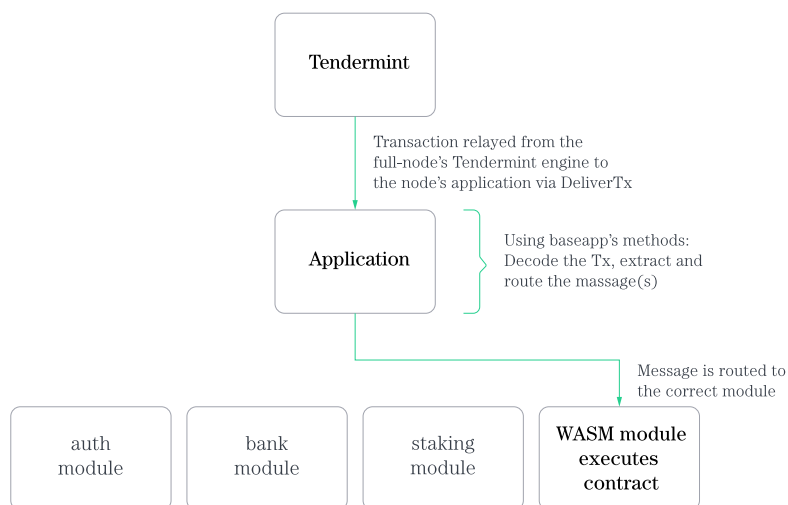
**Upload code:** Once we have compiled the binary, we upload the optimized wasm code, no state nor contract address exists at this stage.

**Instantiate contract:** Instantiate a code reference with some initial state and creates the address which identifies the contract. For example, if you were creating a new ERC-20 token, in this stage you would set the token name, symbol, etc.

**Execute contract:** Each actor has exclusive access to its own internal state. This may support many different calls, but they are all unprivileged; usage of previously instantiated contracts depends on the contract design.

## 5.5 Cosmos SDK Wasm Module

The Coreum blockchain will have a custom module, which will be in charge of processing wasm related messages in order to upload , instantiate and execute smart contracts.



## 5.6 Rust Language

Although you could write WASM smart contracts in many different languages. Rust is the preferred language, it is ideal for generating WASM smart contracts since it has the capability of being memory safe, fast and producing lightweight byte-code for on-chain storage.

# 6 Decentralized Exchange (DEX)

The Decentralized Exchange is a native, built-in exchange functionality within the Coreum blockchain. Although by using smart contracts it is possible to achieve swapping or trading, Coreum aims to build this feature directly in the blockchain core system to allow for a low-fee, secure and fast trading activity with support for all modern trading features.

The DEX can facilitate trading of any issued asset as well as CORE within its function. Users of the Blockchain can choose any asset as base and quote pairs and a market will automatically be created for such pairs. The dex features a full featured native orderbook.

The decentralized exchange will be based on an on-demand orderbook, meaning it allows users to create orders on any possible pair. An order consists of a currency pair, direction, execution price, size and additional conditions for execution and closing.

## 6.1 On-Demand Orderbook

The decentralized exchange will be based on an orderbook in which users can add new pairs and a market will be automatically created for it. An order consists of a currency pair, token that the user wants to buy, execution price, size and additional conditions of execution and closing. A user that created an order will be able to close it before it is fully executed.

Order example:  
Pair: BTC-USD  
Buy: BTC  
Execution price: 30000 USD  
Size: 100 BTC  
Execution conditions: Fill or Kill, Good till time (1 hour)

On decentralized exchange users will be able to trade all the issued assets including the CORE Token and their own issued tokens.

## 6.2 Direct & Indirect Order Matching

In order to fulfill and execute an order a matching order with the same or better price must exist in the opposite direction of the order. For example if one wants to buy 1 BTC for 24000 USD then there must exist opposite orders selling 1 BTC for 24000 USD or less. This is called direct order matching. An indirect order matching means that orders from an intermediary pair will be used to fulfill the order. In the previous example, let's say that there does not exist an order selling 1 BTC for 24000 USD or less but there exists 2 orders in 2 different pairs, one selling 1 CORE for 240 USD and another order selling 10 core for 1 BTC. These 3 orders from 3 different pairs can be matched to fulfill all 3 in a single execution. Coreum will support indirect order matching using only CORE as an intermediary asset, which means for each order ABC-XYZ, coreum will check ABC-CORE and CORE-XYZ to see if it can indirectly match the order.

## 6.3 Order Execution

An order will be executed only by an opposite order either directly or indirectly. After a user adds a new order in the order book, the system finds the opposite order with a matching price which must be equal or better than the order price, and they match with each other. Then users receive funds in the amount calculated from their orders. When two orders match, they will be executed by the price of the order that was created earlier.

## **6.4 Order Types**

Basically they are 2 main types of orders which are the backbone of any orderbook; market orders and limit orders.

### **6.4.1 Market Orders**

A market order is an order in which the order is fulfilled with the best possible opposite orders, regardless of their price.

### **6.4.2 Limit Orders**

Unlike market orders, there is a price limit on which orders can be used to fulfill the current order. Which means that a limit order will be fulfilled using the best possible opposite orders which are the same or better than the price specified in the limit order.

## **6.5 Advanced Features**

### **6.5.1 Good till Cancel Orders**

Orders of this type will be active until the user manually closes the order or the order will be filled with the opposite orders.

### **6.5.2 Good till Time Orders**

Users can set a time limit for orders of this type. 'Good till' time orders can be executed only during the time period entered by the user. When the indicated time period ends, the order will be automatically closed if it was not filled with the opposite orders.

### **6.5.3 Immediate or Cancel Orders**

Users can create an order that must be executed immediately or it will be closed. This order will be executed as much as there are opposite orders matching it until it is fully executed, or will be canceled if there are no more opposite matching orders. This order will not sit as a limit order in the order book.

#### 6.5.4 Fill or Kill Orders

Order with Fill or kill conditions can be executed only entirely at a time. For orders of this type there can be also added a time limitation during which this order can be executed.

## 7 Decentralized Apps (dApps)

Since Coreum is using WebAssembly, it's opening a new corridor for Decentralized App developers, and DeFi applications by allowing them to write smart contracts with their favorite programming language.

Coreum is taking the initiative to help and grow the WASM smart contract developers community and as such has 10% of the total supply of CORE tokens designated for grants to developers.


## 8 Use Cases

Coreum provides developers and financial institutions with a complete essential infrastructure to build any DeFi applications. Moreover, Coreum will incentivize qualified developers to build intuitive dApps. Some of the proposed use-cases of Coreum blockchain are:

- Tokenized Securities (e.g. Sologenic.com)
- Liquidity Providers (LPs) and Market Makers
- Cross-border Payments, Banking and Remittance (e.g. Swift)
- Stablecoin Ecosystems (e.g. USDC, USDT, ...)
- Lending Platforms (e.g. Blockfi, Nexo)
- Wrapped Cryptocurrencies (e.g. ERC20, BEP20)
- Decentralized Exchanges (e.g. Sologenic.org, UniSwap, ...)
- Metaverse Applications (e.g. Decentraland, The Sandbox, Meta)
- NFT Marketplaces (e.g. Sologenic.org, Oopensea.io, ...)
- Gaming and Play-to-earn apps (e.g. Axie Infinity)



## 9 Token Economy

Blockchain Name	Coreum
Token Symbol	CORE
Icon	
Initial Supply	500,000,000

## 10 Allocation

Funds	Percentage	Allocation	Proportion	Vesting Period/ Distribution Plan
Community	70%	SOLO Community Airdrop	20%	1 - 12 months Distribution Schedule
		CORE Community Airdrop	30%	12 - 36 months Distribution Schedule
		Validators' Rewards Pool	10%	Unlocked
		Community d'Ap Developers	10%	Unlocked
Operation	30%	Coreum Maintenance, Operations, Developers, Teams and Investors	30%	1 - 24 months Vesting Period

## 11 References

[1] What is Tendermint

<https://docs.tendermint.com/v0.35/introduction/what-is-tendermint.html>

[2] Tendermint

<https://docs.tendermint.com/v0.35/>

[3] IBC Protocol

<https://ibcprotocol.org/>

[4] Tendermint: Consensus without Mining

<https://tendermint.com/static/docs/tendermint.pdf>

[5] Cosmos SDK gov module

<https://docs.cosmos.network/v0.45/modules/gov/>

[6] Smart contracts introduction

<https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html>

[7] Smart contract security

<https://github.com/ethereumbook/ethereumbook/blob/develop/09smart-contracts-security.asciidoc>

[8] Actors: A Model of Concurrent Computation in Distributed Systems

<https://dspace.mit.edu/handle/1721.1/6952>