Coreum: An Enterprise Grade Blockchain Technical White Paper

Version 2.0 - February 2023

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, Smart Tokens, and NFTs.

Table of content

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 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 ⅓ 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 Interoperablity 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 or unlocking 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.

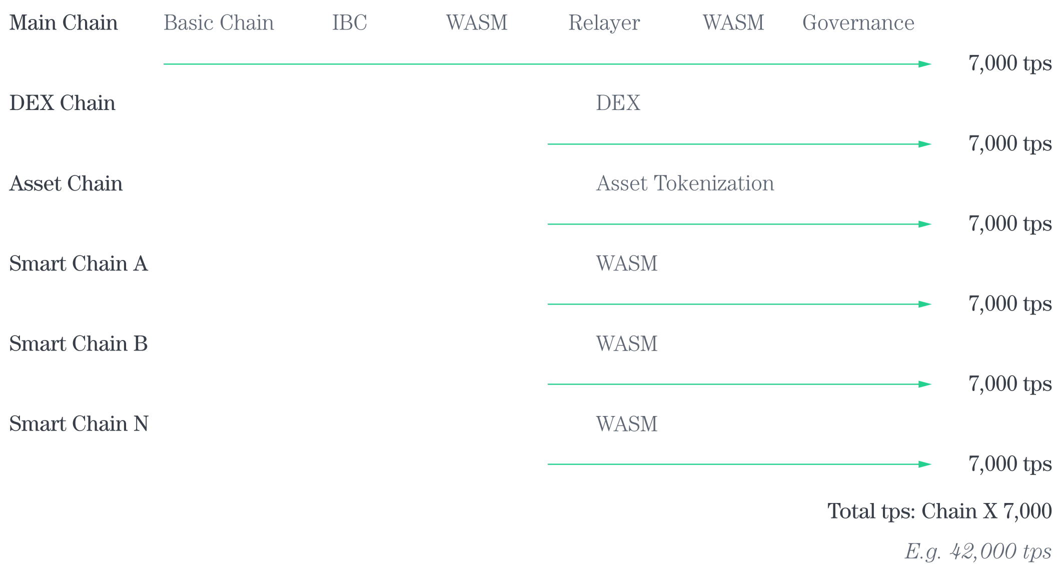
Coreum will implement communication layers either via IBC or bridge to communicate with the following blockchains with more to follow down the line:

- Bitcoin
- Ethereum
- Ripple
- Solana
- Binance Chain & Binance Smart Chain
- Cosmos Hub (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 stakeholders lock their tokens and are granted voting power in exchange, power that can be used to vote on each block. The assumption is that it is to 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, 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 to or higher than the target value then the inflation rate goes down to its minimum and if it is not, it 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 tokens 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 at 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] and their website [2].

Everyone may create a validator by bonding the minimum self-delegated stake required by the network (20,000 CORE tokens). But it does not necessarily mean that the validator will become a part of the consensus. Only 32 validators with the highest stake form the consensus. This mechanism incentivizes network parties to bond more tokens, making the TVL higher, bringing security to the network.

At the genesis block, preconfigured validators will have some voting power allowing the blockchain to start.

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 Validator Rewards

Block rewards will be split between active validators, proportionally to their stakes, 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 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 the target TVL (and it may even go down to 0).

2.5 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 behaviour; 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 block's voting until some time has passed and they un-jail 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 misbehaviour (i.e not showing up to vote on blocks)
2. Counterfactual signing misbehaviour (e.g double signing, rejecting a valid block, etc ...)

The liveness of misbehaviour is taken to be unintentional. In this case, the validator is slashed and put into jail to prevent it from taking part in consensus and being slashed again. After a defined period of time, the maintainer may unjail the validator bringing it back to the consensus process.

On the other hand, the counterfactual signing is assumed to be a malicious one, meaning that the validator is slashed much more aggressively. Moreover, this validator is tombstoned, meaning that it cannot rejoin the consensus anymore.

3 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:

$$fee = gasPrice * gas$$

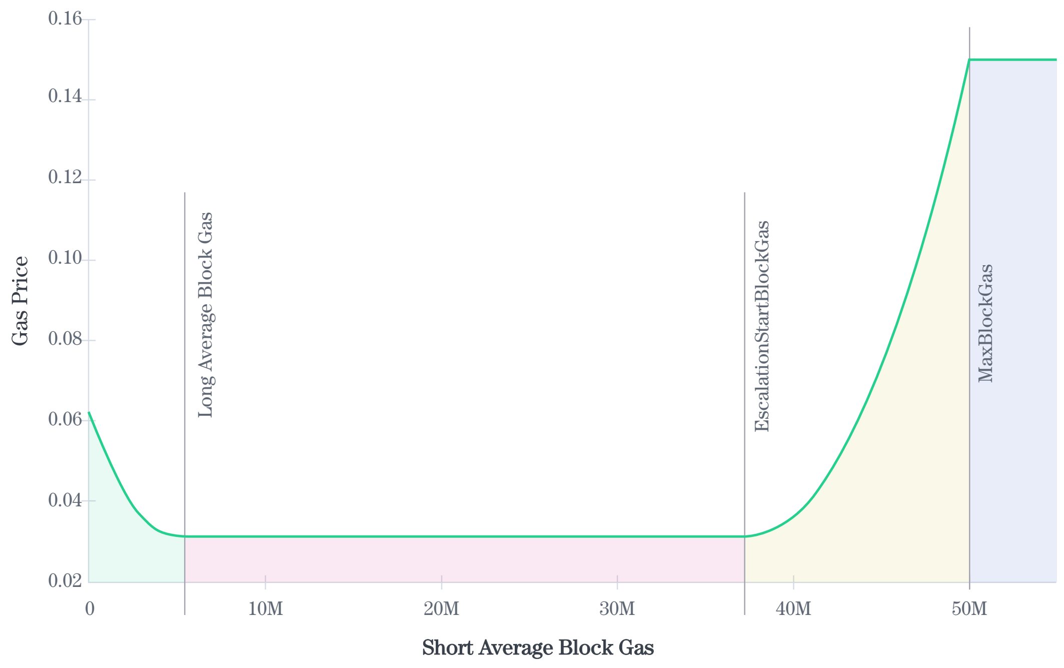
Gas is a measure of how much computational power each transaction needs. The gas variable is invariant for almost all the supported transaction types. Gas consumed by a transaction calling a smart contract may slightly differ between executions.

The gasPrice parameter may change frequently, depending on the load, which will result in having different fees at different times for that transaction.

3.1 Fee Model

Coreum implements the mechanism of automatic gas price adjustment depending on the current transaction load handled by the chain. It is called “fee model”. All the parameters of the fee model are managed via on-chain governance.

The figure below demonstrates the relationship between gas price and gas consumed by the latest blocks:



Axis “x” is defined as “Short Average Block Gas”. This is the moving average computed after every block, based on the gas used by some number (it is a parameter) of the latest blocks.

3.1.1 Green Region

If there are no transactions being executed for some time (or blockchain has just been started) The “Short Average Block Gas” is 0. At that point we set gas price to the value defined by the “Initial Gas Price” parameter.

Then, once transaction load grows, the gas price drops until it reaches the minimum defined by “Maximum Discount”. The maximum discount is reached when “Short Average Block Block Gas” is equal to “Long Average Block Gas”. Long average is computed the same way, the only difference is the much larger number of past blocks being considered.

3.1.2 Violet Region

In this region gas price keeps at the minimum unless the block load reaches a point (“EscalationStartBlockGas”), close to the moment where full capacity of the blockchain throughput is utilized.

3.1.3 Yellow Region

Once we get closer to the saturation point, gas price escalates rapidly to prevent potential attackers from blocking the chain from processing the legit transactions.

3.1.4 Blue Region

In this region full capacity of the chain, defined by "MaxBlockGas" parameter, has been reached. Gas price is set to constant high level until the number of transactions drops back to safe values.

4 Governance

Coreum's Proof of Stake (BPoS) consensus mechanism enables an innovative on-chain governance ecosystem that empowers stakeholders to vote on important decisions aimed at upgrading and improving the chain over time. This approach to governance sets Coreum apart from blockchains like Ethereum, which do not offer any form of on-chain governance.

By involving the community in key decisions, Coreum ensures that its chain is always evolving to meet the needs of its users. A wide range of proposals can be submitted for voting through the on-chain governance system, including fee model parameters, slashing, staking, delegation, number of validators, block times, grants, and more.

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].

5 Smart Tokens

5.1 Introduction

Smart tokens are natively issued tokens on the Coreum chain that are wrapped around smart contracts. They are highly customizable and are designed to be lightweight and flexible.

These tokens exist on the chain's storage and memory, hence, interacting with them does not require calling smart contract functions. We can refer to Smart Tokens as objects and classes that inherit a set of characteristics and functions from the global definition of a token. All tokens have a set of features that are known to the chain. Developers can extend the set of provided functionality and add non-deterministic smart contract-like functions to achieve greater flexibility when developing specific use cases into a token.

Regardless of their type, all tokens share common functions such as minting, sending, burning and so on. The list of default functions is picked from the real use cases of the current DeFi systems and as the chain progresses this list will expand to support more features.

Currently, the following set of features is available for all Smart Tokens:

- Issuance (Minting)
- Access Control List (ACL)
- Burning
- Freezing and Global freezing
- Whitelisting
- Blacklisting
- Burn Rate
- Send Fee
- IBC (Inter Blockchain Communication) Compatibility
- Smart Contract integration

Upon token issuance, the issuer must configure the behaviour of the token and set specific flags that will determine which functions can be triggered at a later stage. These flags are set using the ACL and are as follows:

- can_mint
- can_burn
- can_whitelist

- can_blacklist
- can_partial_freeze
- can_global_freeze
- can_send
- token_transferrable_using_ibc

These flags remain immutable after token issuance and cannot be changed at a later stage. Depending on the token type, these flags can be used to customize what asset needs to be represented by the token. For example, Stablecoins, Crypto, NFT, Stocks, CBDCs and so on.

Once a token is issued and depending on the flags, network participants such as issuers or users can interact with these tokens using the features provided. It is imperative to understand the importance of natively issued tokens and default functions in terms of speed, cost-efficiency, predictability and security, and extendibility.

5.1.1 Speed

Interacting with smart contract functions can often be challenging due to their dependence on the parent smart contract's execution, which can be unpredictable. However, with, the code execution is known to the chain, providing predictability and stability to transactions. This makes sending smart tokens faster and more efficient, without the need to deal with the uncertainties associated with smart contract execution. The predictability of natively issued tokens ensures that transactions on the Coreum chain are fast, reliable, and secure.

5.1.2 Cost-efficiency

Interacting with Smart Tokens costs much less than interacting with Smart Contracts. The fees are always set according to a "known" computational complexity. These transactions are no longer dependent on the amount of gas offered by the caller, but rather are fixed, resulting in a more robust and predictable interaction and management.

When fees are always known, users such as institutions, governments and other DeFi applications can predict how to handle batches of transactions. In addition, the Coreum blockchain offers a discount for bulk transaction submissions, similar to how SaaS-based APIs work with a fee per API call and usage.

5.1.3 Predictability

When execution time, cost and responses are known for a transaction, users can develop much more robust, bug-free and stable applications. Responses from the chain, whether successful or unsuccessful are known and can be handled properly by the developers.es.

5.1.4 Security

One of the main aspects of issuing tokens natively on the Coreum Blockchain is security. When using Smart Contracts, a developer must audit the code of the contract to ensure there are no security vulnerabilities. Over the past few years, hundreds of exploits have been found and abused in Smart Contracts resulting in billions of dollars of losses to the operators and users.

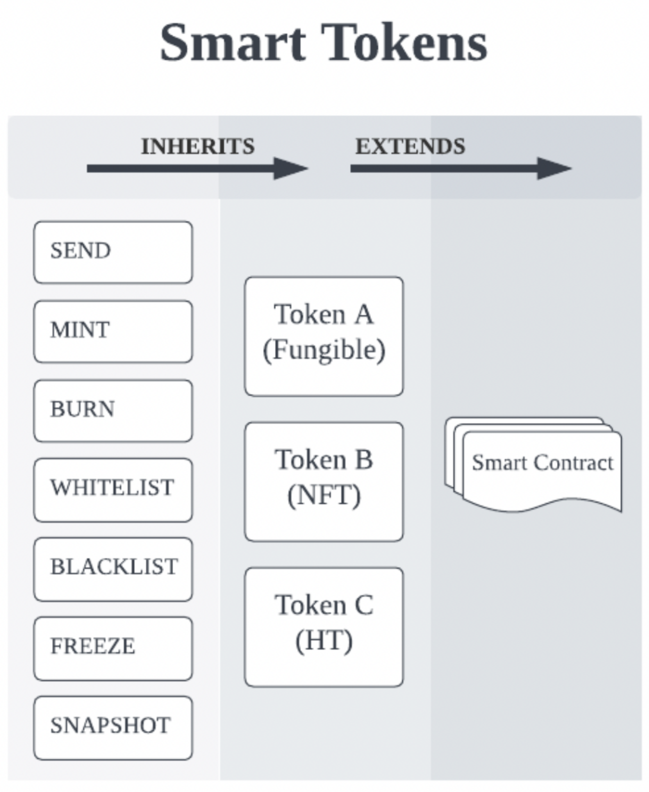
Smart Tokens are not prone to these exploits because the code is known to be the same for all tokens on the chain. The code for the main implementation is audited several times and is open by the public as open-source code to be inspected and audited. Although, an extension to the Smart Token which comes in the form of a smart contract must still be audited by the user. But the majority of the current use cases on the Blockchain such as fungible tokens and basic NFTs can remain risk-free.

5.1.5 Extendibility

Smart Tokens can be exposed to smart contracts for greater customization. Other than the basic features such as sending coins and minting which are provided by the chain by default, the smart contract functions attachment can define a set of new logic and features for a token. For example, one can develop dividend functions in a token that represents shares of a company. Or an NFT can be extended to become interactive with the owner, such as an NFT that can act like a game.

Since Coreum uses WASM as its smart contract engine, developers can easily port and add functionalities developed in several languages and attach them to their digital assets.

Figure 1 displays how all Smart Tokens inherit a set of default functionalities and can be extended by custom smart contract logic.



5.2 Features

5.2.1 Issuance

Asset issuance is the initial phase of the asset lifecycle. On issuance, the issuer defines the asset settings, initial amount, allowed features, and default ACL. After the creation, the initial amount will be minted for the provided recipient and sent to the corresponding account.

The allowed features are the asset features, which can be used with the asset but can't be changed after the issuance. Meaning, if you set no features, you won't be able to use any of them.

The ACL in the issuance is the default ALC configuration. The ACL may include all initially defined features or less.

5.2.2 Minting

Token issuance on Coreum offers versatility and flexibility, with the ability to mint tokens during asset instantiation or on-the-fly if the "can_mint" feature flag is enabled at the issuance level. This feature is useful for a variety of use cases, including CBDCs, Stablecoins, Tokenized Securities, Wrapped Cryptocurrencies, and more.

It is important to note that once a token has given up its ability to mint at the time of issuance, it can never mint more tokens, and the total supply of the token will remain constant. This ensures that the total supply of tokens is predictable and known to all participants, providing stability and security to the Coreum ecosystem.

5.2.3 Access Control List (ACL)

The ACL provides a flexible way for asset administration and is the relationships of the chain accounts and allowed features set on the asset issuance. The administrators might be set in the same ACL.

Example of the ACL:

- can_administrate: account1
- can_partial_freeze: account2
- can_mint: account3, account4
- can_burn: all

5.2.4 Burning

When this feature is enabled on a token, then the holders of the token who have the right permission will be able to burn some of the tokens they hold to reduce the total supply of that token. To give an example use case, if shares of a company are tokenized, then the total supply will represent the total shares of the company on the chain. And burning those tokens would mean those shares are now moved out of the blockchain and total supplies will correctly represent that fact.

5.2.5 Freezing

If this feature is enabled when the token is first issued, freezing allows the administrator of the token to freeze a portion of or the balance of the token held by a user. There are many use cases that are enforced by law to freeze a token. An example use case is when the token administrator sends tokens to an address but does not want the user to spend them until some time has passed such as clearing a cheque.

The token might also be frozen globally, meaning, nobody would be able to transfer that token until it is unfrozen back.

5.2.6 Whitelisting

Whitelisting is designed to support scenarios when, due to KYC/AML or any other policy enforced by the issuer, Smart Token might be held by a limited scope of accounts that passed the verification procedure. Its possible use cases include tokens representing stock shares, CBDC or any other rights enforced by the legal system where the identity of the holder must be confirmed.

If the can_whitelist flag is enabled on the Smart Token, the issuer defines the list of accounts (addresses) that are allowed to receive that token. If an account is not on that list and someone tries to send tokens to it, the transfer is rejected. Alternatively, a special flag whitelist_everyone may be set to true to whitelist all the accounts. If whitelisting was enabled during Smart Token issuance, the issuer may update the list of whitelisted accounts and enable or disable the whitelist_everyone flag at any time.

If an account holding the token is removed from the whitelist, its current balance stays unaffected but it cannot receive tokens anymore until whitelisted back.

5.2.7 Blacklisting

The blacklisting feature in Coreum allows issuers to prevent a specific account from receiving or sending a Smart Token. This can be beneficial in instances where a token holder is being investigated for potentially illegal activities, such as money laundering, fraud, or terrorist financing. In these situations, the account can be temporarily blocked until the investigation is completed.

In addition, many regulators require a mechanism for promptly blocking suspicious accounts, and blacklisting is a key component in implementing such measures. This helps bring real-world assets to the blockchain in a compliant and secure manner.

If the can_blacklist flag is enabled on the Smart Token, the issuer may blacklist any account at any time, and it is the issuer's sole decision if and when the account is enabled back. Until that happens, an account can neither send nor receive the token. The balance stored on the account is not affected, meaning that existing tokens are never revoked.

If both whitelisting and blacklisting features are active, blacklisting always takes precedence.

5.2.8 Burn Rate

The burn rate is a parameter that can be specified when a token is issued. It represents a portion of the transferred amount that is destroyed or "burned" during a token transfer. The burn rate is expressed as a number between 0 and 1 and is added to the original transfer amount as a charge to the sender. It is important to note that the burn rate does not apply to transfers made by the token issuer.

5.2.9 Send Fee

Similarly to the burn rate, the issuer may set a sending fee. It is a number between 0 and 1 too, which defines the portion of the transferred amount which is sent to the issuer. This tax is charged to the sender in addition to the original amount.

Send fee is not applied to transfers sent or received by the token issuer.

5.2.10 IBC (Inter Blockchain Communication) Compatibility

Assets are based on the cosmos modules and extend the cosmos IBC capability. Hence they can be transferred to and from IBC-supported chains within the cosmos ecosystem or outside it given proper relayers are present. When an asset is transferred to another chain, it’s represented as a tokenized version of the underlying asset in the Coreum Blockchain.

5.2.11 Smart Contract Integration

An integral part of the Coreum blockchain is to support the deployment and execution of Smart Contracts, which already bring countless possibilities. By implementing Smart Tokens, Coreum is able to extend its functionalities and flexibility through Smart Contracts.

Developers can deploy their own logic by writing Smart Contracts, being able to issue Smart Tokens, mint and burn them, whitelist and blacklist accounts, and freeze and unfreeze their balances.

Now the behavior of the Smart Token might be driven by actions taken by ordinary people, departments of your organization or even different ones cooperating together, meaning that the final action taken on the Smart Token might be the result of the process involving different actors following the transparent logic provided by the Smart Contract, leading to greater reliability.

6 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 the 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:

- Security flaws (re-entrance 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.

6.1 WASM

WASM (WebAssembly) is a highly scalable and efficient engine for executing smart contracts, developed by the W3C in collaboration with major companies like Mozilla and Google. It's portable, Turing-complete, and supports multiple programming languages such as C/C++, C#, Rust, JavaScript, TypeScript, Haxe, Kotlin, and Go.

WASM is fast, efficient, open-source, easy to debug, platform-independent, and memory-safe, making it an ideal choice for executing smart contracts.

6.2 CosmWasm

Since the 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 Cosmo's 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 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.

6.3 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 the 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.

6.4 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.

7 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 the 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 fully functional 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.

7.1 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.

7.2 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.

7.3 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 then 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.

7.4 Order Types

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

7.4.1 Market Orders

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

7.4.2 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.

7.5 Advanced Features

7.5.1 Good till Cancel Orders

GTC orders are active until either the user manually cancels the order or it is filled with matching orders on the market. This means that the order will remain open until one of these two events occurs, ensuring that the user's trade intentions are fulfilled, whether it be through manual cancellation or execution.

7.5.2 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.

7.5.3 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 cancelled if there are no more opposite matching orders. This order will not sit as a limit order in the order book.

7.5.4 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.

8 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.

9 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)
- EGB (ISO 20022) compatibility layer

10 Token Economy

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

11 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

12 References

[1] What is Tendermint
`https://docs.tendermint.com/v0.35/introduction/what-is-tender-mint.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`