Smart Tokens: A Highly Customizable and Cost-effective Solution to Tokenize Complex Fungible, Non-fungible and Hybrid tokens on the Blockchain.

Reza Bashash, Wojciech Malota, Yaroslav Savchuk, Milad Zahedi, Dzmitry Hill,
Coreum Development Foundation

Nov 16, 2022

Abstract The advent of Blockchain technology and, more specifically, Smart Contracts have proposed revolutionary ways for assets to exist in a decentralized manner. Currently, considered a proof of concept, this technology has become interesting for major sectors such as governments and financial institutions.

Each entity is trying to utilize the Blockchain to build real use cases such as Decentralized Finance (DeFi) applications. However, with the current technology and systems in place, it would be extremely inefficient for these actors to rely on non-deterministic transactions.

In this paper, the team proposes a new solution to tokenize different types of assets (fungible, non-fungible and hybrid) in the Coreum Blockchain through the use of native tokens that are extended by Smart Contract functionalities, hence, Smart Tokens.

Introduction

The blockchain was born out of the necessity for a trustless, peer-to-peer payment transaction system that would not rely on a centralized third-party such as governments and banks. Over the years, the blockchain has evolved to provide complex applications using smart contracts. A smart contract is an executable piece of code that runs on top of the blockchain to facilitate, execute and enforce an agreement between parties without the involvement of a trusted third party. [1]
Smart Contracts can represent assets such as fungible tokens (e.g. US Dollar, Bitcoin, ...), non-fungible tokens (NTFs, such as uniquely identifiable objects like art, digital code and blueprints) and hybrid tokens which might act as fungible and non-fungible at the same time.

Through smart contracts, each of these assets can have unique properties and functionalities. For example, for fungible tokens, there exists an ability to mint more tokens, burn existing tokens and transfer tokens between accounts. However, for non-fungible tokens, these functionalities could be extended to provide additional features to objects, such as representing a “ticket” as an NFT that could only be used once, remain transferable or become non-transferrable. Other types of NFTs could be used for KYC verifications (Soulbound NFTs), these would not be able to be transferred to any other accounts and would only be used for digital identity, onboarding and account management.

Such logic needs to be directly programmed in a smart contract and be programmatically called by users by paying fees. Fees in open blockchains (such as Ethereum) that allow smart contract executions are calculated when the program runs and are represented as “gas” units. The Ethereum Virtual Machine is a quasi–Turing-complete state machine, "quasi" because all execution processes are limited to a finite number of computational steps by the amount of gas available for any given smart contract execution. [2] Thus this transaction cost (Gas in the Ethereum blockchain) is not deterministic and it is only possible to estimate it based on the state of the network and other factors such as congestion. A smart contract function could be so complex that it would never get executed, consuming all the gas fees proposed by the caller. Similarly, if the network is congested, the amount of fee consumed by triggering functions within smart contracts could be so high that it would make such action financially unreasonable.

Smart contracts are usually created and distributed by the community of developers through a series of proposed standards such as ERCs (Ethereum Request for Comments), in the Ethereum Blockchain. For instance, the ERC-20 introduces a standard for Fungible Tokens, in other words, they have a property that makes each Token exactly the same (in type and value) as another Token, meaning that 1 Token is and will always be equal to all the other Tokens.

Example functionalities ERC-20 provides:

- *transfer tokens from one account to another*
- *get the current token balance of an account*
- *get the total supply of the token available on the network*
- *approve whether an amount of token from* an account can be spent by a third-party account

If a Smart Contract implements the following methods and events it can be called an ERC-20 Token Contract and, once deployed, it will be responsible for keeping track of the created tokens on Ethereum. [3] Although open token standards such as ERC20s are widely used, they remain limited in terms of adoption by real use cases.

Coreum Blockchain has a different perspective on tokens and assets in general and is introducing a new concept that is called Smart Tokens.

Smart Tokens

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 while remaining extendible.

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*
- *Whitelisting*
- *Blacklisting*
- *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.

Speed: As mentioned earlier, calling functions of smart contracts can be tricky as they rely on the parent smart contract execution which is unknown. However, natively issued tokens are predictable and the code execution is known to the chain. This makes transactions such as sending smart tokens much faster with Smart Tokens.

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.

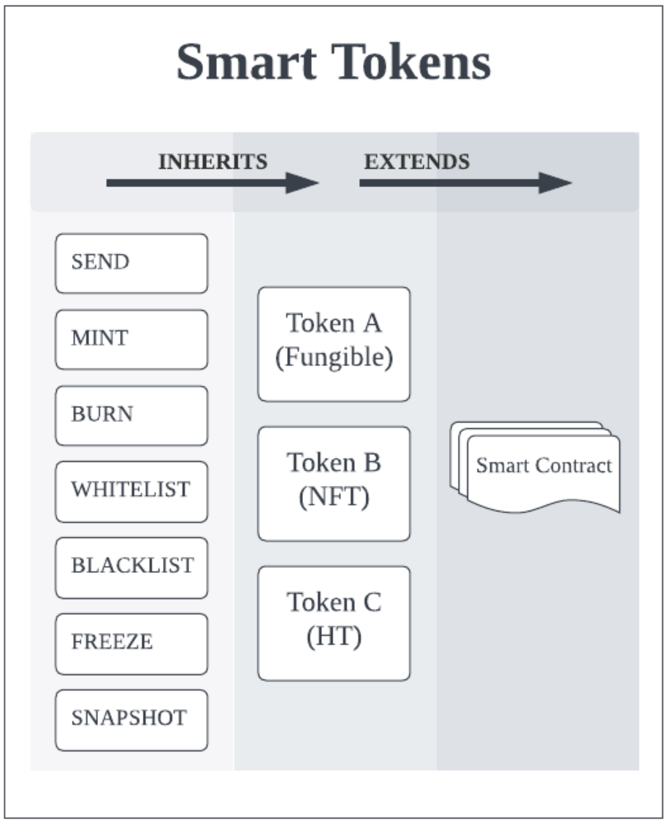
Predictability: When execution time, cost and responses are known for a transaction, users can develop much more robust, bug-free and stable applications. Responses from the chain, whether successful or unsuccessful are known and can be handled properly by the developers.

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.

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.



(Figure 1 - Smart Tokens)

I.Features

A. 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.

B. Minting

Tokens can be initially minted on asset instantiation or on the fly if the can_mint feature flag is set at the issuance level. This feature is useful for a wide variety of use cases, such as CBDs, Stablecoins, Tokenized Securities, Wrapped Crypto Currencies and so on. If a token gives up the ability to mint at the time of issuance, it can never mint more tokens and the total supply of the token will not ever increase.

C. 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

D. 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.

E. 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.

F. 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.

G. Blacklisting

Blacklisting allows the issuer to temporarily or permanently block an account from receiving and sending the Smart Token. This feature might be helpful in situations when an investigation is ongoing against the token holder, who is suspected of money laundering, fraud, terrorist financing etc. In those cases, an account might be blocked until results are collected.
Many regulators require a procedure for blocking suspicious accounts as soon as potential criminal activity is detected. So blacklisting is an essential mechanism required to bring all kinds of real-life assets to the blockchain.

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.

H. 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.

I. Smart Contract integration

Integral part of the Coreum blockchain is support for deploying and executing Smart Contracts. The flexibility offered by Smart Contracts brings countless possibilities. We are connecting Smart Tokens to the flexibility of Smart Contracts, meaning that tokens might be issued and managed by the Smart Contract.

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.

References

1- https://arxiv.org/pdf/1710.06372.pdf
2- https://aly.arriqaaq.com/turing-completenes-ethereum-bitcoin/
3- https://eips.ethereum.org/EIPS/eip-20