hal0x2328 Update README.md with official Twitter account af04c87 a day ago

1 contributor

111 lines (64 sloc) 8.96 KB

Master Contract Token (MCT)

A utility token for the Neo Smart Economy

Scripthash: a87cc2a513f5d8b4a42432343687c2127c60bc3f

Features

- NEP-5 compatible
- Directly tradable by smart contracts without off-chain code
- Ability to pass additional parameters along with a token transfer to a smart contract
- Offers "staked storage" to contracts holding a minimum amount of MCT

Overview

The NEO Smart Economy has a major limitation in that although smart contracts can receive funds (system assets like NEO or GAS) they cannot send funds without intervention of off-chain code that responds to events emitted by the smart contract and creates and broadcasts new transactions.

The introduction of NEP-7 in Neo 2.7.5 has eased some of the difficulties with receiving system assets by a smart contract. Now a contract will be able to be triggered upon receiving NEO/GAS, giving the contract the opportunity to perform actions or reject the transfer. The problem of the inability to send NEO/GAS in a single operation still remains, however.

Current NEP-5 tokens have more limitations than system assets:

- The smart contract will receive no notification that the tokens were sent
- The smart contract will not be able to spend tokens it holds without coordination from an off-chain wallet holding the owner's key

A change is needed to foster automated exchange of value between smart contracts and users. As a result of this need, we have created the Master Contract Token (MCT).

MCT is a NEP-5-compatible token implementation with some additional features. The most notable being the ability for third-party smart contracts to send, receive, and hold it.

Implementation by third-party smart contracts

Receiving MCT tokens via smart contract:

1. The existing NEP-5 `transfer` implementation in MCT is modified so that the scripthash of the recipient of the transfer is checked against all deployed smart contracts on the blockchain.
2. If the scripthash belongs to a smart contract (determined by `GetContract(scripthash)` NEO API call), the `transfer` implementation dynamically invokes the `OnTokenTransfer` operation in that contract.
3. The `OnTokenTransfer` operation informs the third-party smart contract that it has been delegated to receive a certain balance of MCT tokens.
4. `OnTokenTransfer` can be written to perform any function during its execution phase, and returns True if the token transfer should be finalized, False otherwise. In the case that a smart contract does not have support for receiving tokens, invoking the `onTokenTransfer` operation should fail, also preventing the transfer from occurring and protecting against user error (e.g. pasting the wrong recipient scripthash/address).
5. Depending on the return value from the third-party smart contract, the MCT NEP-5 implementation finalizes the transfer.
6. Optionally, the invoker can overload the `transfer` call with additional arguments over and above the required `from`, `to`, and `amount` for the receiving contract's `OnTokenTransfer` operation to use. This allows for more capability in the receiving contract's code with only minor changes to existing wallet software.

Sending MCT tokens from a smart contract:

1. The third party smart contract invokes the `transfer` operation of the MCT NEP-5 contract with its own scripthash as sender, along with the recipient scripthash and the amount to be transferred.
2. The MCT NEP-5 `transfer` implementation detects that the transfer is coming from an address that corresponds to a known smart contract, and instead of using `CheckWitness(scripthash)` to verify the owner of the funds, it verifies that the calling contract's scripthash matches the sender's.

Any future NEP-5 contracts could operate in the same manner with the above mentioned changes to the `transfer` operation.

Any third party contract that wishes to receive and hold these types of tokens merely needs to implement the `OnTokenTransfer` functions as described above in their application (`Application` trigger type) code.

Additionally, the third-party contract owner may want to add a function to transfer tokens to the owner's wallet on demand, once `CheckWitness` has been used to verify the owner’s signature, if the tokens are ever to be redeemed for some other currency (in the case of an ICO that accepts MCT as payment, for example).

Staked storage

Third-party smart contracts that stake a minimum amount of MCT tokens are allowed to use the MCT smart contract as a storage proxy. The minimum starting stake is 10,000 (ten-thousand) MCT, and can be lowered by the contract owner over time if the value of MCT rises significantly. The minimum stake can never be raised, only lowered. Once the contract has staked tokens, as long as the current minimum stake is maintained, the proxy storage for that contract will work.

Storage keys are prefixed with the scripthash of the calling contract before all standard storage operations (`Put`, `Get` or `Delete`). This ensures separation of data between different users of staked storage, however it reduces the maximum key length to 1000 bytes.

Using staked storage in the MCT contract means a smart contract author can deploy a contract requiring basic storage capabilities for 90 GAS instead of 490 GAS - a substantial savings.

Advanced features such as `Find` and `Migrate` are not implemented. If a smart contract needs these features, it is advisable to pay the 490 GAS for a storage-enabled contract rather than use staked storage.

Example code

An example contract for a dApp that uses MCT as its means of exchange of value as well as for storage can be found at https://github.com/Splyse/MCT/blob/master/mct-dapp-template.py - this contract could be deployed for only 90 GAS.

TestNet contract

Two versions of MCT exist on the Neo TestNet, tokens CTX (scripthash 9aff1e08aea2048a26a3d2ddbb3df495b932b1e7) and CTY (scripthash 81d9cbc994dd104d3e03514b47eb23c6c406b2e7). CTX was airdropped to TestNet addresses holding TestNet NEP-5 tokens but additional tokens may be requested by developers looking to test their contracts against the TestNet tokens.

Privnet contract

The MCT contract source is not yet publicly available, however, a custom-compiled version is available for the neo-privnet-docker environment, using the private key of the privnet owner wallet (address: AK2nJJpJr6o664CWJKi1QRXjqeic2zRp8y, WIF: KxDgvEKzgSBPPfuVfw67oPQBSjidEiqTHURKSDL1R7yGaGYAeYnr) as the contract owner. Download from https://github.com/Splyse/MCT/blob/master/mct-privnet.avm

Frequently Asked Questions

When ICO?

There will never be an MCT ICO, the MCT smart contract *is* the final product. An initial airdrop was used to distribute MCT into the ecosystem. There will be no further giveaways. Anyone pretending to be someone from the MCT team claiming they are giving away MCT at this point is fraudulent.

What were the airdrop rules?

Tokens were distributed to any Neo address that received a transfer of *any* existing NEP-5 token on the Neo MainNet before May 8, 2018 00:00:00 UTC (ending with MainNet block 2240482). 567,442,000 tokens were airdropped to 404,453 Neo addresses.

1,000 tokens were distributed for each third-party token held for some length of time, regardless of the amount.

For example, if a Neo address held only RPX, RHT and ONT (in any quantity) at any point before May 8, 2018, they would have received an airdrop of 3,000 MCT.

This airdrop distribution is designed to put MCT into the hands of the most active users participating in the NEP-5 token ecosystem rather than just the richest holders.

I have NEP-5 tokens in an exchange wallet, will I get the airdrop?

That is up to the exchange to add MCT and credit your account with the airdrop. With exchanges, the bottom line is - if you do not hold the private keys to a wallet, it is not really your wallet.

What is the distribution and supply of MCT?

580,000,000 tokens is the maximum supply of MCT and 100% of that has already been distributed. Approximately 98% of these were distributed to the community in the airdrop. The remaining ~2% of tokens were allocated to the smart contract owner (Splyse) to offset development and deployment costs and to provide ongoing support for users of the token.

Is there a website?

https://github.com/Splyse/MCT is the only official website. Beware of phishing pages offering tokens in exchange for private keys.

Is there a Discord/Telegram server?

There is a Discord server at https://discord.gg/vrA8tcq. There is no Telegram server or any other official support forum.

Scam alerts

Twitter

Twitter username @mcttoken is *not* us - we do not know who is behind this account although it is pretending to be the official Twitter feed of MCT. Beware of attempts to steal your private keys or funds by scammers promising to send you MCT - as we said - any such giveaways are bogus. The airdrop is over and there will be no further distributions from the team. The official MCT twitter account is https://twitter.com/MCT_Token