
Minime



build passing

The MiniMeToken contract is a standard ERC20 token with extra functionality:

The token is easy to clone!

Anybody can create a new clone token from any token using this contract with an initial distribution identical to the original token at a specified block. The address calling the `createCloneToken` function will become the token controller and the token's default settings can be specified in the function call.

```
1 function createCloneToken(  
2     string _cloneTokenName,  
3     uint8 _cloneDecimalUnits,  
4     string _cloneTokenSymbol,  
5     uint _snapshotBlock,  
6     bool _isConstant  
7 ) returns(address) {
```

Once the clone token is created, it acts as a completely independent token, with it's own unique functionalities.

Balance history is registered and available to be queried

All MiniMe Tokens maintain a history of the balance changes that occur during each block. Two calls are introduced to read the totalSupply and the balance of any address at any block in the past.

```
1 function totalSupplyAt(uint _blockNumber) constant returns(uint)  
2  
3 function balanceOfAt(address _holder, uint _blockNumber) constant  
   returns (uint)
```

Optional token controller

The controller of the contract can generate/destroy/transfer tokens at its own discretion. The controller can be a regular account, but the intention is for the controller to be another contract that

imposes transparent rules on the token's issuance and functionality. The Token Controller is not required for the MiniMe token to function, if there is no reason to generate/destroy/transfer tokens, the token controller can be set to 0x0 and this functionality will be disabled.

For example, a Token Creation contract can be set as the controller of the MiniMe Token and at the end of the token creation period, the controller can be transferred to the 0x0 address, to guarantee that no new tokens will be created.

To create and destroy tokens, these two functions are introduced:

```
1 function generateTokens(address _holder, uint _value) onlyController
2
3 function destroyTokens(address _holder, uint _value) onlyController
```

The Token's Controller can freeze transfers.

If `transfersEnabled == false`, tokens cannot be transferred by the users, however they can still be created, destroyed, and transferred by the controller. The controller can also toggle this flag.

```
1 // Allows tokens to be transferred if true or frozen if false
2 function enableTransfers(bool _transfersEnabled) onlyController
```

Applications

If this token contract is used as the base token, then clones of itself can be easily generated at any given block number, this allows for incredibly powerful functionality, effectively the ability for anyone to give extra features to the token holders without having to migrate to a new contract. Some of the applications that the MiniMe token contract can be used for are:

1. Generating a voting token that is burned when you vote.
2. Generating a discount "coupon" that is redeemed when you use it.
3. Generating a token for a "spinoff" DAO.
4. Generating a token that can be used to give explicit support to an action or a campaign, like polling.
5. Generating a token to enable the token holders to collect daily, monthly or yearly payments.
6. Generating a token to limit participation in a token sale or similar event to holders of a specific token.
7. Generating token that allows a central party complete control to transfer/generate/destroy tokens at will.
8. Lots of other applications including all the applications the standard ERC 20 token can be used for.

All these applications and more are enabled by the MiniMe Token Contract. The most amazing part being that anyone that wants to add these features can, in a permissionless yet safe manner without affecting the parent token's intended functionality.

How to deploy a campaign

1. Deploy the MinimeTokenFactory
2. Deploy the MinimeToken
3. Deploy the campaign
4. Assign the controller of the MinimeToken to the campaign.