Achain ATP1.0 Token Standard

Preamble

> Title: ATP1.0 Token Standard
>
> Author: Arthur Qiang
>
> Type: Standard

Simple Summary

A standard interface for tokens.

Abstract

ATP1.0（Achain Token Protocol） standard allows for the implementation of a standard API for tokens within smart contracts. This standard provides basic functionality to transfer tokens, as well as allow tokens to be approved so they can be spent by another on-chain third party.

Parameter format:

NOTE: Smart contract method parameters uniform format：Parameter1|Parameter2|Parameter3

Use "|" to separate the parameters

Parameter Type: String

E.g. transfer_to(ACTD2iQ9A25cZ9rEtPwd7BNAyimRhyaYTdS6|2999.98000|XXXX)

Smart contract internal variable explanation（Must be initialized）:

contractName	Token name
officalAddress	Address
symbol	Token symbol
decimals	Token decimals
totalSupply	Token totalSupply

E.g. Modify the color of the place， To complete the Token.（Token symbol：XXX）

```
local function initContractInfo()
    local info:ContractInfo = ContractInfo()
    info.contractName="XXX_COIN"
    info.officalAddress="ACTD2iQ9A25cZ9rEtPwd7BNAyimRhyaYTdS6"
    info.symbol="XXX"
    info.decimals = 100000
    info.totalSupply=10000000000000000
    return json.dumps(info)
end

function M:COIN_XXX()
end
```

Methods：

contractInfo

Returns the Basic Information of the token.

E.g.

```
function M:contractInfo()
Return：{"symbol":"XXX","decimals":100000,"totalSupply":1000000000,"contractName":"XXX_TEST","officalAddress":"ACTJHPo83uhSkD1tQLyt1fx4Duh5NKSD5HEY"}
Events:  emit contractInfo({"symbol":"XXX","decimals":100000,"totalSupply":1000000000,"contractName":"XXX_TEST","officalAddress":"ACTJHPo83uhSkD1tQLyt1fx4Duh5NKSD5HEY"})
```

contractName

Returns the name of the token.

E.g.

```
function M:contractName()
Return：XXX_TEST
Events: emit contractName(XXX_TEST)
```

symbol

Returns the symbol of the token.

E.g.

```
function M:symbol()
Return：XXX
Events: emit symbol(XXX)
```

officalAddress

Returns the official address of the token.

E.g.

```
function M:officalAddress()
Return：ACTD2iQ9A25cZ9rEtPwd7BNAyimRhyaYTdS6
Events: emit officalAddress(ACTD2iQ9A25cZ9rEtPwd7BNAyimRhyaYTdS6)
```

decimals

Returns the number of decimals the token uses - e.g. `100000`, means to divide the token amount by `100000` to get its user representation.(Now only support to 5 decimal places)

E.g.

```
function M:decimals()
Return：100000
Events: emit decimals(100000)
```

totalSupply

Returns the total token supply.

E.g.

```
function M:totalSupply()
Return：1000000000000000  (totalSupply * decimals)
Events: emit totalSupply(1000000000000000)
```

balanceOf

Returns the account balance of another account with address `_owner`.

E.g.

```
function M:balanceOf(_owner:string)
Return：100000000
Events: emit balanceOf(100000000)
```

transfer_to

Transfers `_value` amount of tokens to address `_to`, and must fire the `transfer_to_success or transfer_to_fail` event.

E.g.

```
function M:transfer_to(_to_value:string)
Return：true/false
Events: emit transfer_to_fail('Illegal parameter')
    Or: emit transfer_to_success(ACT6YVz3LevfwuY1pANWtm7G52UcDZC7AFr:63351989410,ACTD2iQ9A25cZ9rEtPwd7BNAyimRhyaYTdS6:24799975201,2931,1512125670)
        Format：（_from:balance,_to:balance,version（Increasing）,time（Time Stamp））
```

transferFrom

Transfers `_value` amount of tokens from address `_from` to address `_to`, and must fire the `transfer_to_success or transfer_to_fail` event.

The `transferFrom` method is used for a withdraw workflow, allowing contracts to transfer tokens on your behalf. This can be used for example to allow a contract to transfer tokens on your behalf.

E.g.

```
function M:transferFrom(_from_to_value:string)
Return：true/false
Events: emit transfer_to_fail('Illegal parameter')
    Or: emit transfer_to_success(ACT6YVz3LevfwuY1pANWtm7G52UcDZC7AFr:63351989410,ACTD2iQ9A25cZ9rEtPwd7BNAyimRhyaYTdS6:24799975201,2931,1512125670)
        Format：（_from:balance,_to:balance,version（Increasing）,time（Time Stamp））
```

approve

Allows `_spender` to withdraw from your account multiple times, up to the `_value` amount. If this function is called again it overwrites the current allowance with `_value`.

E.g.

```
function M:approve(_spender_value:string)
Return：true/false
Events: emit approve_fail('Illegal parameter')
    Or: emit approve_success({"balance":10000000,"to_address":"ACTRkFMvPB9Ahc9iLzL3xJsJLpYiJro6J52","from_address":"ACT7XcjJExK6iFXYyM6F49vvGFYB6P6Xkyu9"})
```

allowance

Returns the amount which `_spender` is still allowed to withdraw from `_owner`.

E.g.

```
function M:allowance(_owner_spender:string)
Return：100000000
Events: emit allowance_fail('Illegal parameter')
    Or:  emit allowance_success(100000000)
```

COIN_XXX

Compatible with old smart contracts，XXX is Token symbol.

```
Nothing
```

Events:

E.g.

```
emit transfer_to_fail('Illegal parameter')
```