eosio.token contract

cc32d9 / Europechain

eosio.token contract

- https://github.com/EOSIO/eosio.contracts/tree/master/contracts/eosio.token

Symbol

- Currency symbol is a combination of precision and 1-7 capital letters for symbol name: “4,XEC” means the minimum amount is 0.0001 XEC.
- Symbol is encoded as uint64.
- In token contract, symbol is used as primary key in **`stat`** and **`accounts`** tables.

eosio.token.hpp

- Action definitions: **`create`**, **`issue`**, **`transfer`**, …
- Table definitions: **`accounts`**, **`stat`**

create()

Issuer will be allowed
to call issue()

Only contract account is
allowed to create a new symbol

Stats table has symbol as scope

```
void token::create( const name&   issuer,
                    const asset&  maximum_supply )
{
    require_auth( get_self() );

    auto sym = maximum_supply.symbol;
    check( sym.is_valid(), "invalid symbol name" );
    check( maximum_supply.is_valid(), "invalid supply");
    check( maximum_supply.amount > 0, "max-supply must be positive");

    stats statstable( get_self(), sym.code().raw() );
    auto existing = statstable.find( sym.code().raw() );
    check( existing == statstable.end(), "token with symbol already exists" );

    statstable.emplace( get_self(), [&]( auto& s ) {
       s.supply.symbol = maximum_supply.symbol;
       s.max_supply    = maximum_supply;
       s.issuer        = issuer;
    });
}
```

Symbol is also the primary key

issue()

The action increases the issuer’s balance and adds it to total supply

```
void token::issue( const name& to, const asset& quantity, const string& memo )
{
    auto sym = quantity.symbol;
    check( sym.is_valid(), "invalid symbol name" );
    check( memo.size() <= 256, "memo has more than 256 bytes" );

    stats statstable( get_self(), sym.code().raw() );
    auto existing = statstable.find( sym.code().raw() );
    check( existing != statstable.end(), "token with symbol does not exist, create token
before issue" );
    const auto& st = *existing;
    check( to == st.issuer, "tokens can only be issued to issuer account" );

    require_auth( st.issuer );
    check( quantity.is_valid(), "invalid quantity" );
    check( quantity.amount > 0, "must issue positive quantity" );

    check( quantity.symbol == st.supply.symbol, "symbol precision mismatch" );
    check( quantity.amount <= st.max_supply.amount - st.supply.amount, "quantity exceeds
available supply");

    statstable.modify( st, same_payer, [&]( auto& s ) {
       s.supply += quantity;
    });

    add_balance( st.issuer, quantity, st.issuer );
}
```

transfer()

```
void token::transfer( const name&    from,
                      const name&    to,
                      const asset&   quantity,
                      const string&  memo )
{
    check( from != to, "cannot transfer to self" );
    require_auth( from );
    check( is_account( to ), "to account does not exist");
    auto sym = quantity.symbol.code();
    stats statstable( get_self(), sym.raw() );
    const auto& st = statstable.get( sym.raw() );

    require_recipient( from );
    require_recipient( to );

    check( quantity.is_valid(), "invalid quantity" );
    check( quantity.amount > 0, "must transfer positive quantity" );
    check( quantity.symbol == st.supply.symbol, "symbol precision mismatch" );
    check( memo.size() <= 256, "memo has more than 256 bytes" );

    auto payer = has_auth( to ) ? to : from;

    sub_balance( from, quantity );
    add_balance( to, quantity, payer );
}
```

Normally the sender is paying for recipient's RA
to store the token balance

Amount is atomically subtracted from sender
and added to recipient balance

Other actions

- `retire`: issuer can destroy tokens and remove them from total supply
- `open`: receiver can open a zero balance and pay for RAM, so that sender doesn’t have to pay for it
- `close`: owner of zero balance can release the RAM. Older versions of token contract deleted zero balance automatically.

Design considerations

- When receiving a payment, always check the contract name! There are few dozens of token contracts with EOS symbol.
- Some contracts extend eosio.token code and implement additional logic within the token contract. A recommended approach is to leave most of additional logic outside of token contract where possible.