## Comments

* one SC per country
* citizen ID number: IDN
* company number: CN

## ID and Addresses

The addresses will be hashes of IDN’s and CNs.

struct wallet {

string IDN/CN;

address hashOfIDN;

uint balance;

bool valid; (true: valid, false: invalid)

}

mapping citizensWallets: hashOfIDN (uint, I think) => wallet

mapping companiesWallets: hashOfIDN (uint, I think) => wallet

## Transactions

* **verifyIDCitizen**Description:  
  Inputs:
* **verifyCompanyNumber**Description:  
  Inputs:
* **creationOfCitizenWallet**Description**:** create a wallet and add it to citizensWallets

Inputs:

* **creationOfCompanyWallet**

Description: create a wallet and add it to companiesWallets

Inputs:

* **validationOfCitizensWallets**Description**:**

Inputs:

* + list of wallets to validate
* **invalidationOfCitizensWallets**Description**:**

Inputs:

* + list of wallets to invalidate
* **validationOfCompaniesWallets**Description**:**

Inputs:

* + list of wallets to validate
* **invalidationOfCompaniesWallets**Description**:**

Inputs:

* + list of wallets to invalidate
* **transferFromCitizenToCompany**

Description: Allow transfer of toenailed from individual account to company account

Inputs:

* + @ company address
  + Amount to pay

Checks:

* + Is @ a company address ?
  + Does Tx. Init a individual @ ?
  + Does Tx. Init have enough fund ?
* **sendRedeem**

Description : Commercial @ can call this function to mark X token they have as redeemed . One way function ?

Inputs:

* **sendFundsAsTax**Description:  
  Inputs:
* **burnFunds**Description: Since we want to incentivize people to spent the money they received and not hoard it. By calling this function, we will burn % of unspent token for each individual @  
  Inputs:

## Roles

* ISSUER - can create addresses and fund them
* BURNER - can call the burnFunds function
* CITIZEN
* COMPANY
* ADMIN