ShardEx documentation
Free TON wallet as a Chrome extension

1. About

This document represents RSquad's submission for Free TON wallet as a Chrome extension contest.

The code of the developed Free TON wallet as a Chrome extension can be found at the following link: https://github.com/RSquad/ShardEx

1.1. Contacts

telegram: @jackkru69, @inyellowbus

2. Glossary

- Multisig is an abbreviation for multisignature – multiple signature or a combination of several key signatures used to confirm and execute transactions. This technology is used in multisignature wallets.

- Wallet address – unique address of the wallet on the blockchain. It explicitly identifies the wallet and is required for any actions with the wallet to be performed. It does not, on its own, provide anyone access to wallet funds. Wallet custodian – authorized owner of the wallet. Owns the private key and corresponding seed phrase, which are required to make any changes to the wallet or wallet funds. Wallet may have more than one custodian.

- SafeMultisig – basic multisignature wallet, does not permit contract code modification. Is required if you use validator scripts.

- SetcodeMultisig – more advanced multisignature wallet.

- SetcodeMultisig2 – more advanced multisignature wallet. This version is currently required to create a wallet that can be managed in TON Surf.

3. Overview

The implementation of the System requirements from the conditions of the contest is described in the table below:

3.1. Generic

Feature	Status	Comment
English language of the interface;	Fully supported	
Support of Google Chrome;	Fully supported	
Absence of analytical trackers (Google Analytics, Yandex Metrika, etc.);	Fully supported	
Support of mainnet and testnet(s);	Fully supported	
On-chain activity history (transactions, messages, contract interactions, etc.);	Fully supported	
Any calls that require the user’s keys must ask for the password input to decrypt them from the local storage.	Fully supported	

3.2. Wallet features

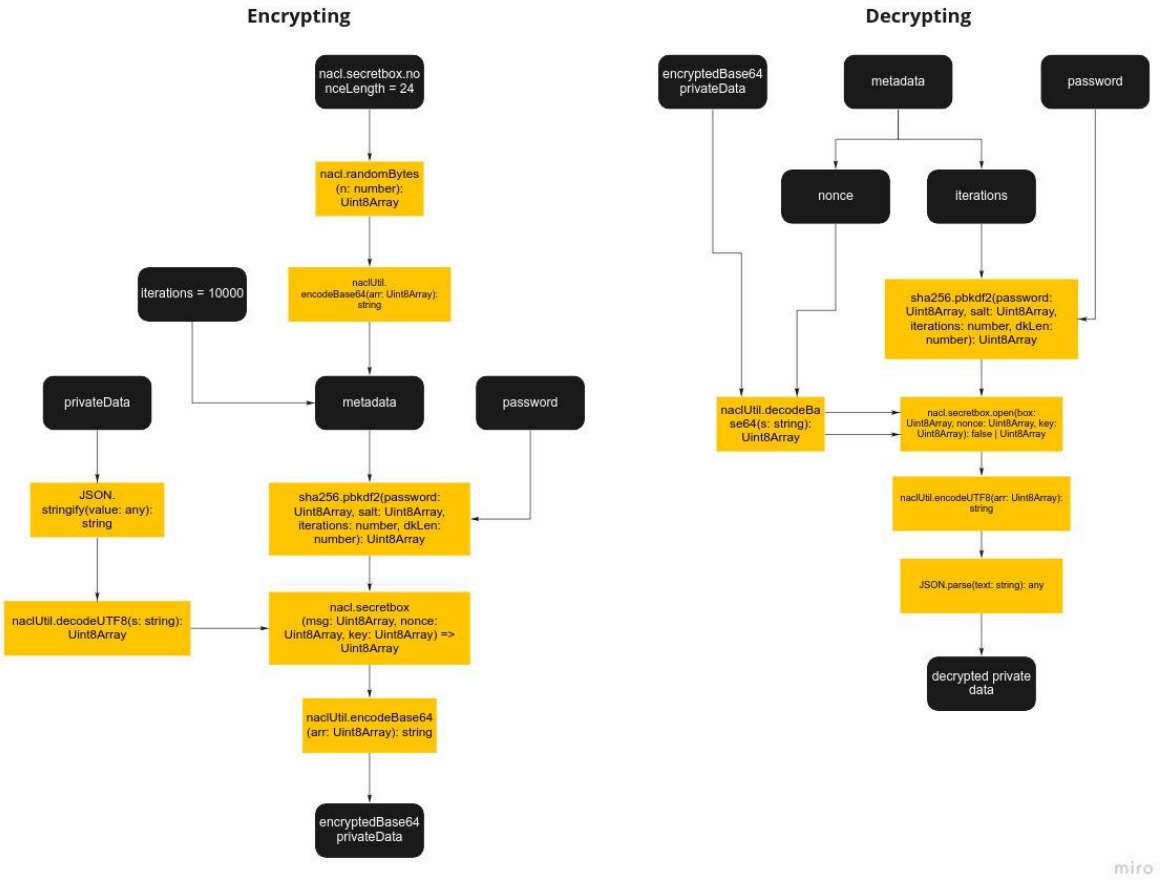
Feature	Status	Comment
Native support of any open-sourced non-custodial Free TON wallets, e.g.: - Original TON wallets (Wallet v. 3); - TON Labs' wallets (SafeMultisig, SetCodeMultisig);	Partially supported	does not support Original TON wallets (Wallet v. 3);
Random seed phrase generation;	Fully supported	
12 or 24 words wallet initialization (based on wallet contract);	Fully supported	
Wallet seed phrase backup and restoration;	Fully supported	
Public and private keys generation, backup, and restoration;	Fully supported	
Encrypted local key storage;	Fully supported	
Password protection;	Fully supported	
Support of sending a memo with messages (or encoded payload).	Fully supported	

Some additional features that are not described in the contest stage 1 conditions are shown in the table below:

Feature	Status	Comment
Multisig wallet		Creating multisign wallet (many owners, propose transaction, confirm transaction)
Full-screen mode		
Multi accounts		
Change password		

3.3. Security

All data private data is encrypted using tweetnacl's xsalsa20-poly1305 implementation. The encryption key is derived from the password using PBKDF2/SHA256. The iteration count for the PBKDF2 invocation is configurable and defaults to 10,000 rounds.

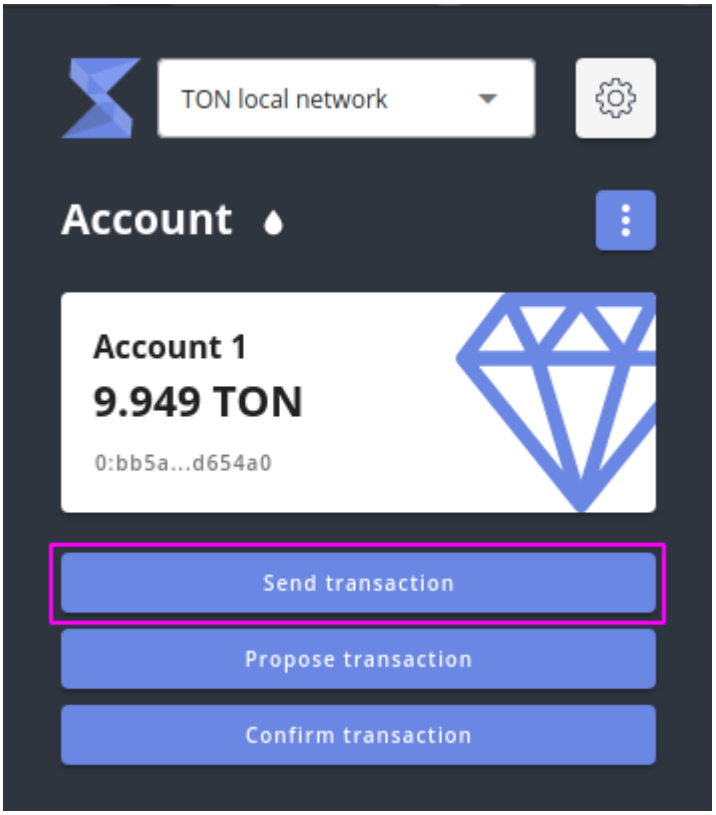


4. Free TON wallet extension

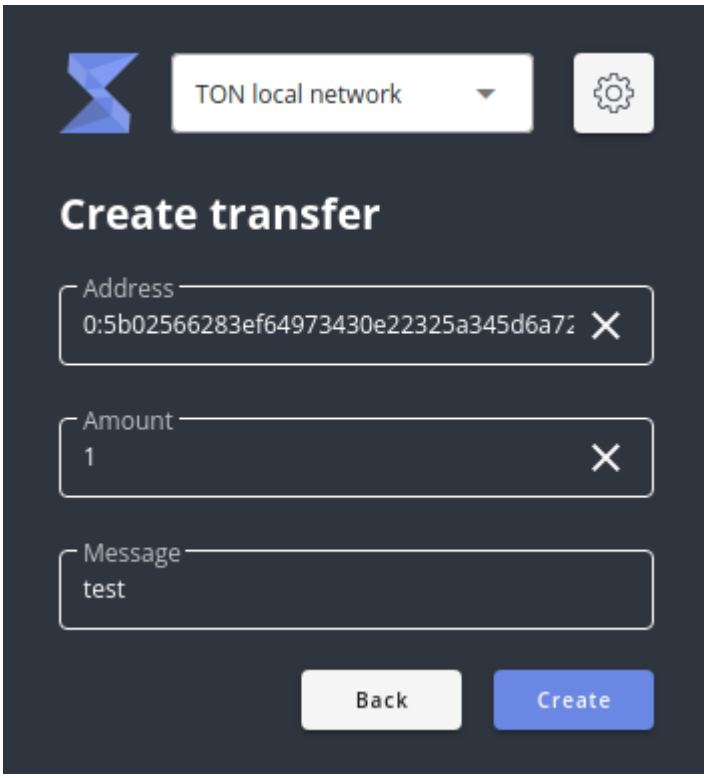
4.1. First launch of the add-on

4.2. Transfer

1. Open Popup
2. Enter the lock password
3. Click the Send transaction button



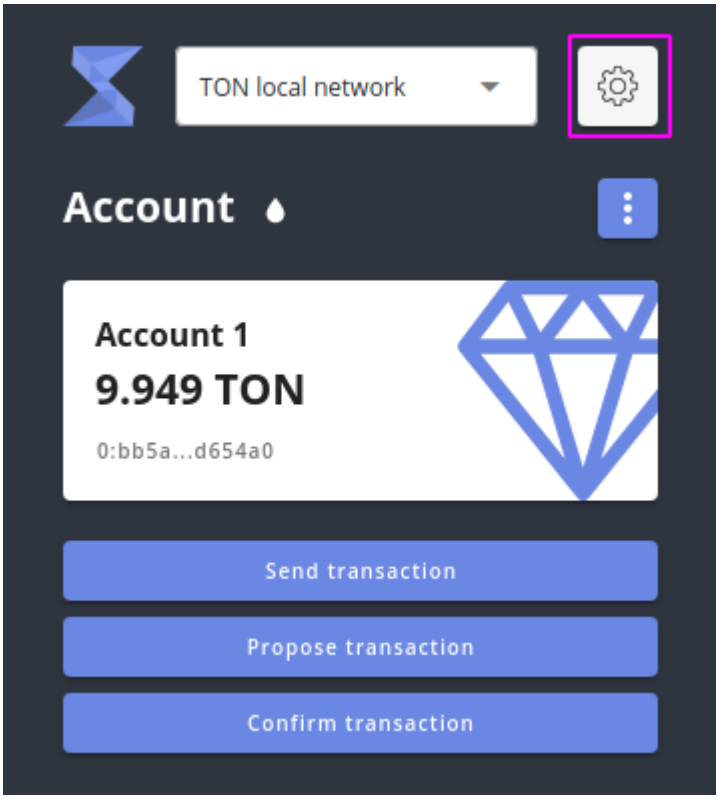
4. Enter the wallet address
5. Enter the amount
6. Enter a comment (optional)



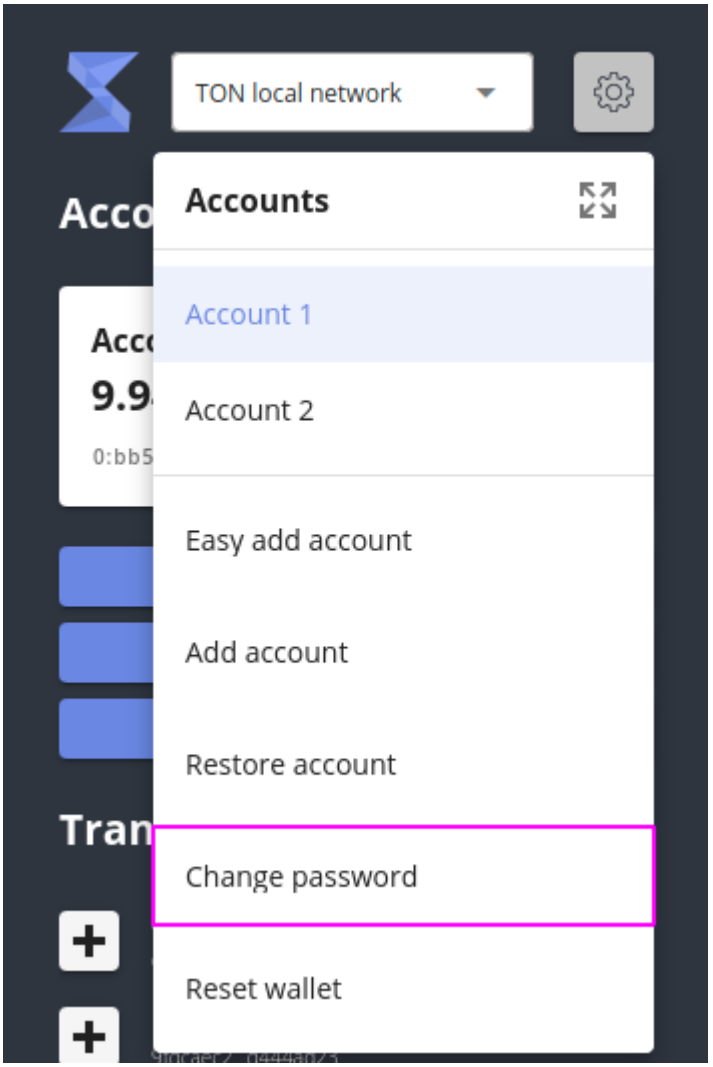
7. Click the Create button

4.3. Change password

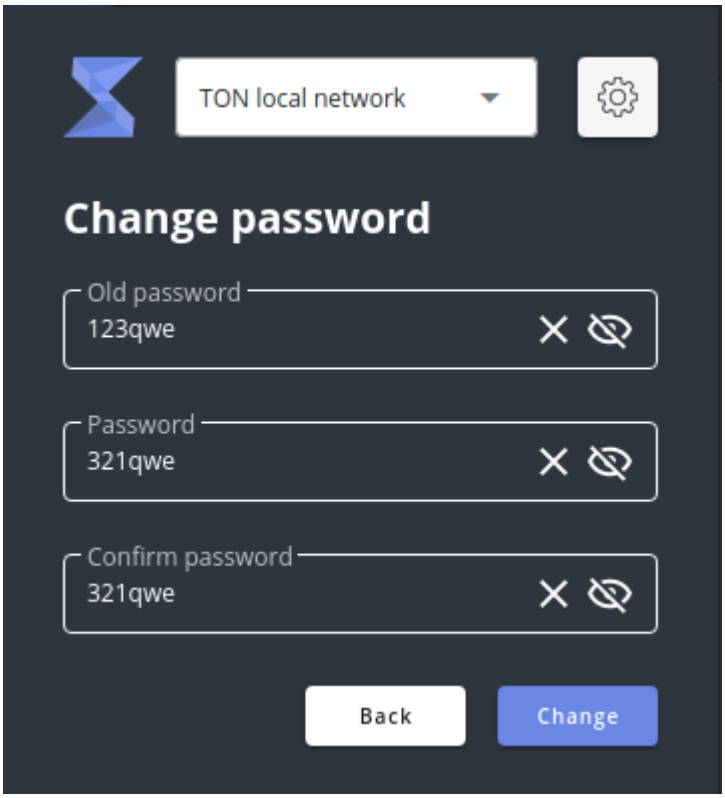
1. Open Popup
2. Enter the lock password
3. Click on the main menu



4. Click Change password



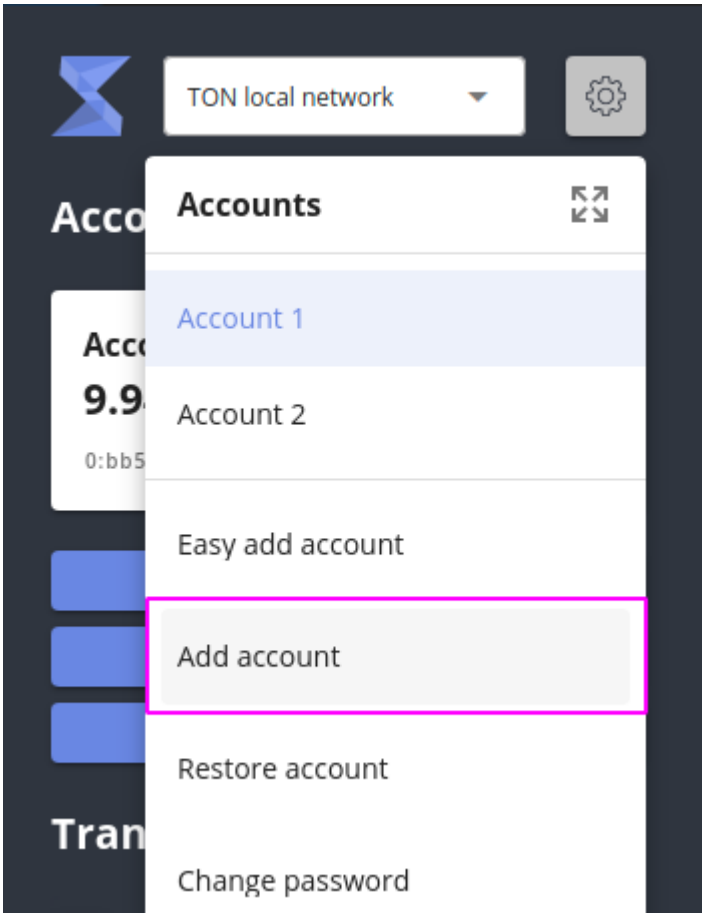
5. Fill out the form



6. Click the Change button

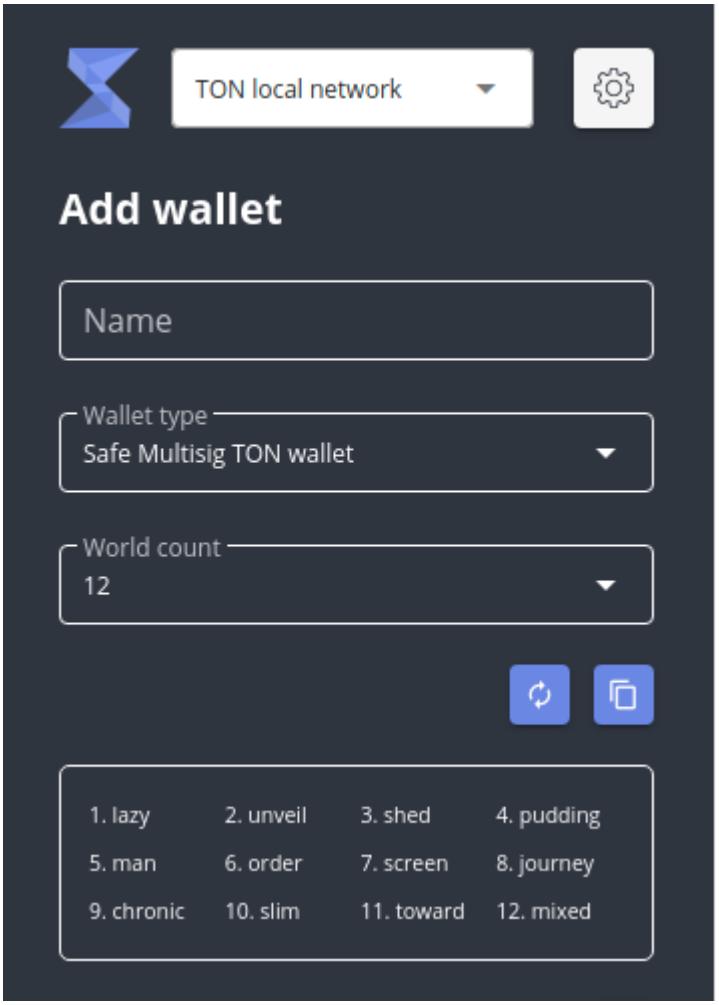
4.4. Add account

1. Open Popup
2. Enter the lock password
3. Click on the main menu
3. Click the Add account button

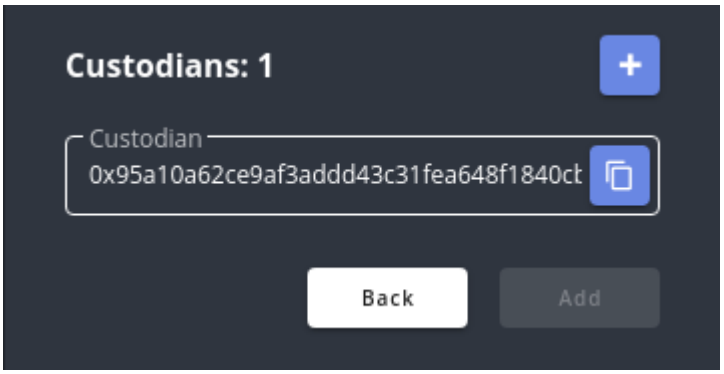


4. Enter a name, select the type of wallet and the number of words

5. Write or copy seed phrase



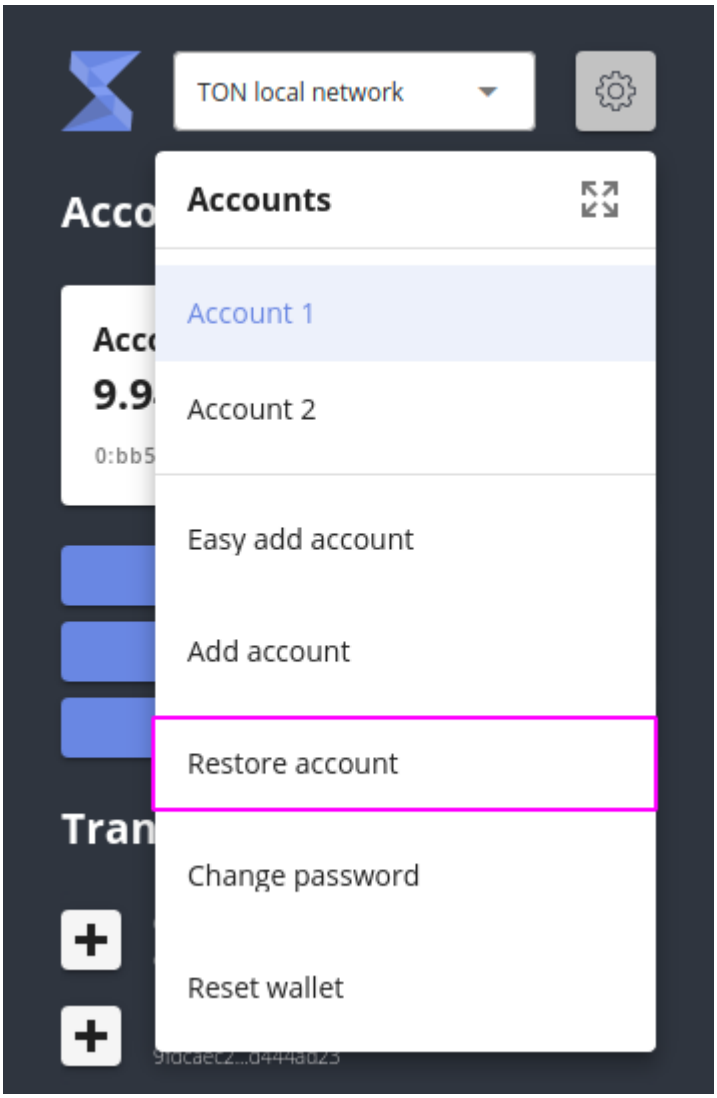
6. Enter other owners or leave yourself



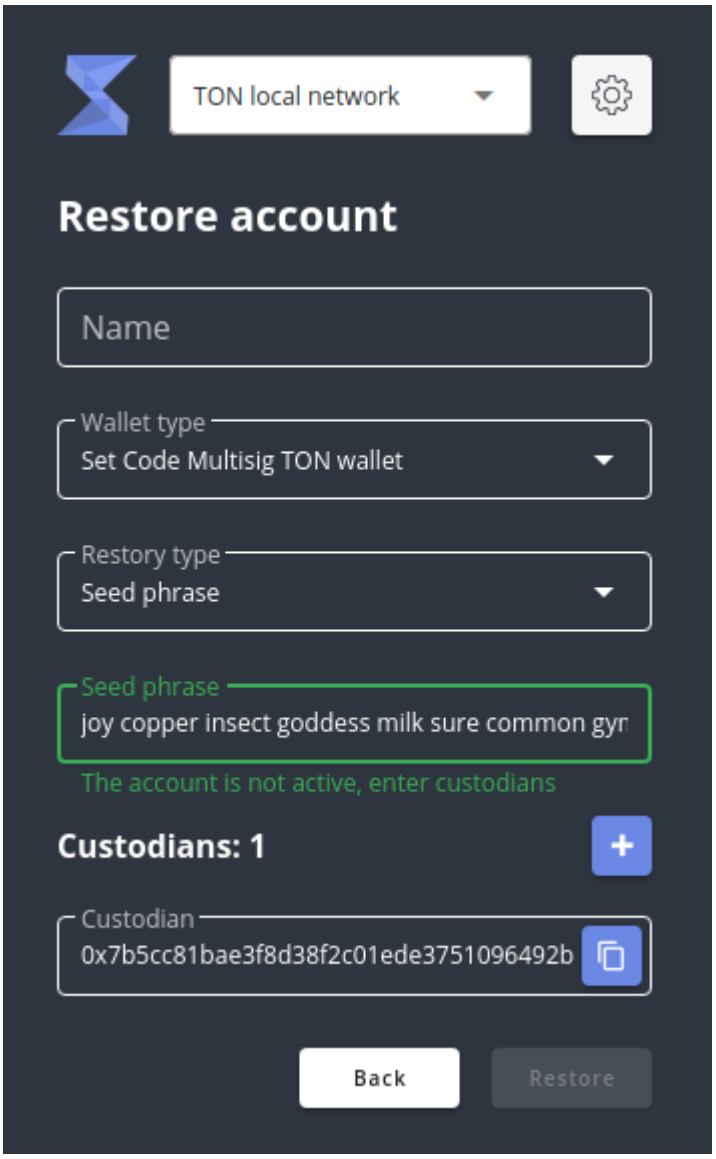
7. Click the Add button

4.5. Restore account

1. Открыть Popup
2. Enter the lock password
3. Click on the main menu
4. Click Restore account



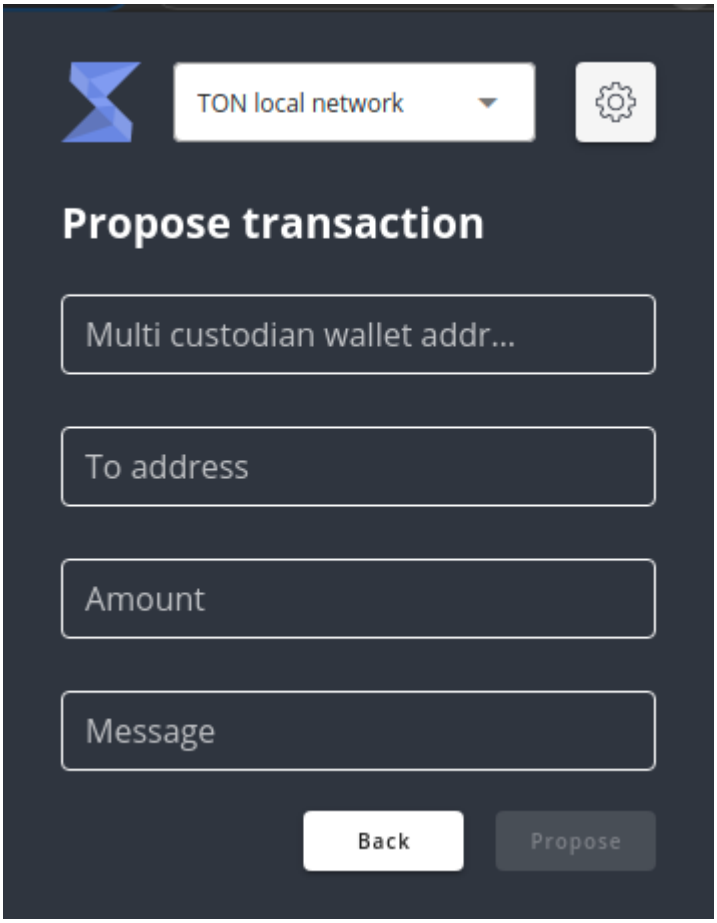
5. Enter a name, select the wallet type, and the recovery type (using seed phrase or keypair)
6. Enter seed phrase or keypair
7. If the account is not deployed enter the owners



8. Click the Restore button

4.6. Propose transaction

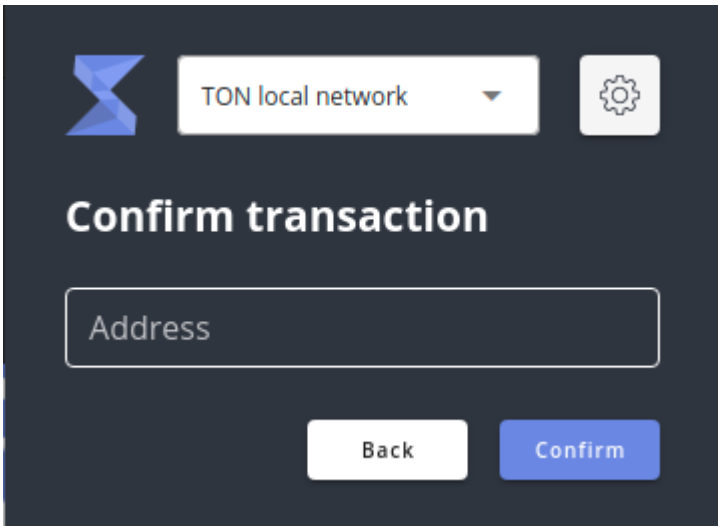
1. Open Popup
2. Enter the lock password
3. Click Propose transaction



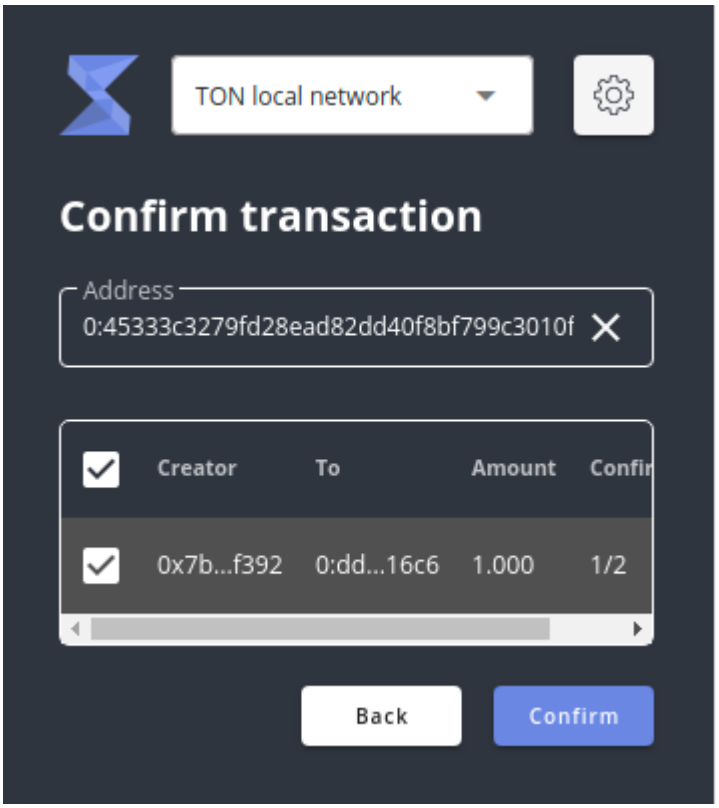
4. Enter the shared wallet address
5. Enter the recipient's wallet address
6. Enter the amount and comment(optional)
7. Click the Propose button

4.7. Confirm transaction

1. Open Popup
2. Enter the lock password
3. Click button Confirm transaction



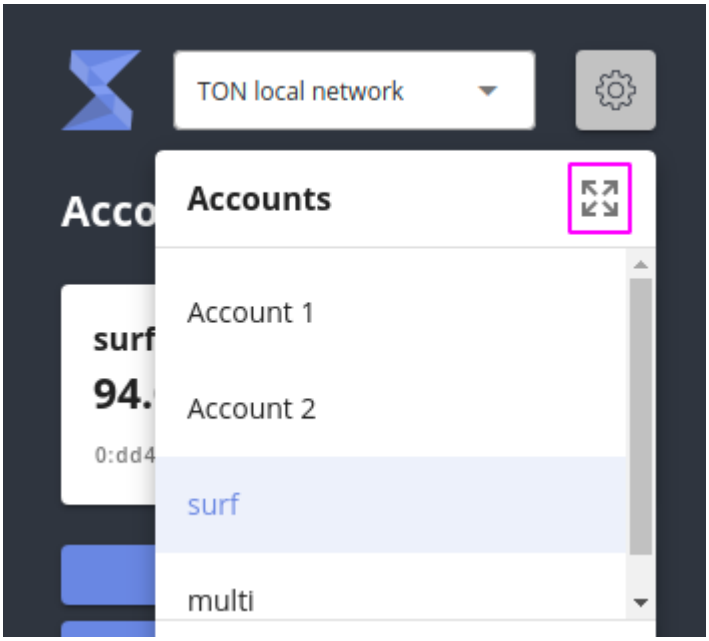
4. Enter the shared wallet address
5. Select transactions to confirm



6. Click the Confirm button

4.8. Full-screen mode

1. Open Popup
2. Enter the lock password
3. Click on the main menu
4 Click Expand View icon



5. Data view

5.1. Keystore

```
interface KeyMetadata {
  nonce: string;
  iterations: number;
}

export type Key = {
  metadata: KeyMetadata;
  public: string;
  private: string;
};

You, 3 weeks ago | 1 author (You)
class KeystoreState {
  keys: any = {};
}
```

5.2. Accounts

```
export type WalletType = "safe-multisig" | "set-code-multisig" | "set-code-multisig2";

export type TokenType = {
  name: string;
  symbol: string;
  balance: string;
  decimals: number;
};
You, 6 days ago | 1 author (You)
export interface AccountInterface {
  address: string;
  walletType: WalletType;
  name: string;
  custodians: string[];
  publicKey: string;
  tokens: TokenType[];
  networks: string[];
  isExist: boolean;
  isRestoredWithKeyPair: boolean;
}
You, a month ago | 1 author (You)
class AccountsState {
  accounts: AccountInterface[] = [];
}
```

6. Technical stack

extension: https://github.com/Kocal/vue-web-extension
ui: https://github.com/vuetifyjs/vuetify
extension storage: https://github.com/championswimmer/vuex-persist +
https://developer.chrome.com/docs/extensions/reference/storage/

7. Build and Deployment

7.1. Overview

ShardEx application is built using vue-web-extension (https://github.com/Kocal/vue-web-extension)

npm run build
Build the extension into dist folder for production.

A zip file is also built and is located in artifacts directory.

npm run serve
Build the extension for development and watch over file changes.

It also automatically reload your extension into your browsers, thanks to webpack-extension-reloader plugin.

For a local network, installation is required tondev (https://github.com/tonlabs/tondev)

Prerequisites

- `Node.js` >= 10.x installed
- (optional) `Docker` >= 19.x installed
- Solidity compiler requires VC++ Runtime on Windows. You can install it from the latest supported Visual C++ downloads.

Install

```
npm i -g tondev
```

If you see an EACCES error when you try to install a package globally on Mac or Linux, please see this instruction

Then `tondev se start`

7.2. Deploying as browser extension

1. Open the Extension Management page by navigating to `chrome://extensions`.
    - The Extension Management page can also be opened by clicking on the Chrome menu, hovering over **More Tools** then selecting **Extensions**.
2. Enable Developer Mode by clicking the toggle switch next to **Developer mode**.
3. Click the **LOAD UNPACKED** button and select the extension directory.