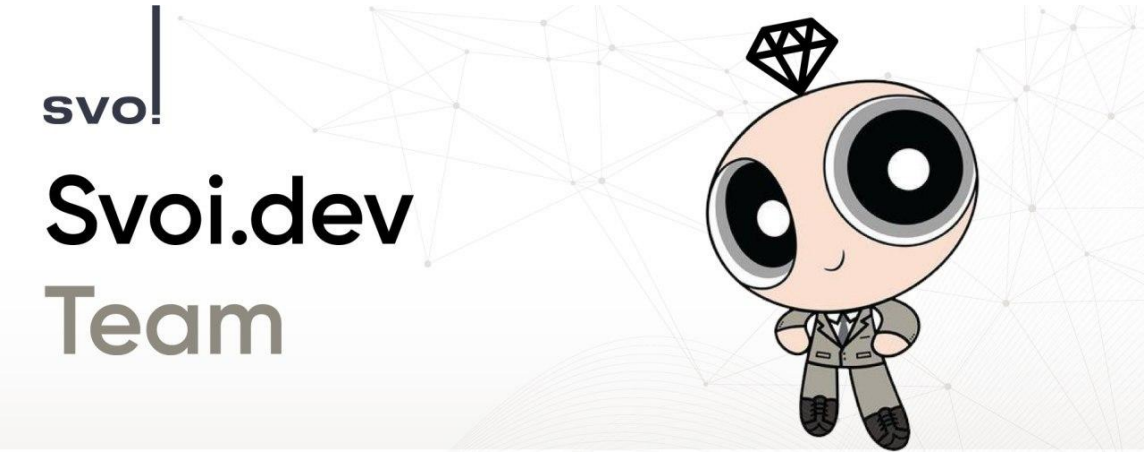


TONWallet

Artifacts

Github: https://github.com/SVOIcom/browser-extension
Site: https://tonwallet.io
0.0.2 release: https://github.com/SVOIcom/browser-extension/releases/tag/0.0.2

Contacts

- TG: @svoidev @lailune / e-mail: support@tonwallet.io
- Wallet: 0:0439186aa0147661ebaf2b32ecc76bac172fcdaa24c7df7c9cb03cc816e435e6

Submission criteria checking

Conformity with the hard criteria:

- English language of the interface - (Supports English, Russian, Spanish)
- Support of Google Chrome; - (Support all Chromium-based browsers)
- Absence of analytical trackers
- Support of mainnet and testnet(s); - (Also support custom networks)
- On-chain activity history;
- Any calls that require the user's keys must ask for the password input to decrypt them from the local storage.
- Native support of any open-sourced non-custodial Free TON wallets - (Includes SURF and SafeMultisig contract. Can work with all SafeMultisig-like contracts)
- Random seed phrase generation
- 12 words wallet initialization
- Wallet backup and restoration
- Public and private keys generation, backup, and restoration
- Encrypted local key storage
- Password protection

- Support of sending a memo with messages - (and encoded payload within API)

Conformity with the soft criteria:

- Multilanguage support - (Supports English, Russian, Spanish. Can be extended by adding language files)
- The extension is published in the Chrome store (Chrome Web Store in progress, Microsoft Edge add-ons on progress)
- Support of additional browsers (All chromium-based browser)
- Brevity
- Mostly everyday English to facilitate understanding
- Readiness to participate in the implementation of the solution in the next stage - :)
- Verifiable extension security along with the process to verify the equality of published version with source code (Extension written with Google Chrome extensions safety guide https://developer.chrome.com/docs/extensions/mv3/security/ and uses well-documented, modular code)

More valuable futures:

- TIP3 tokens support, transfer, receive and creation
- Multi wallet contracts support
- Custom network support
- Multiaccounts support
- Dark and light mode - depends on browser settings
- Custom account names
- DeNs support

Extension summary

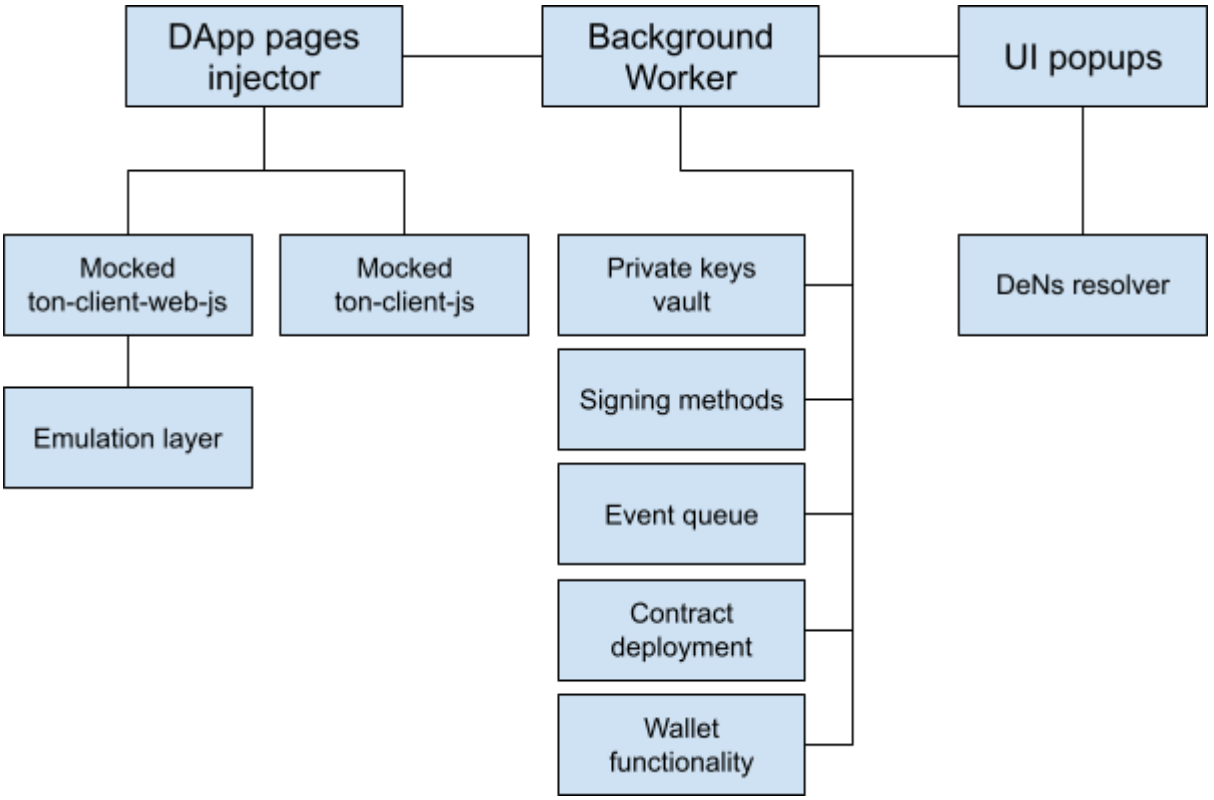
TONWallet - browser extension. Provides two main features:

1. Creation and management of FreeTON wallets and TIP3 tokens wallets
2. Providing FreeTON network connection to web-page with decentralized application

Main features:

1. Work with the main and test FreeTON network, and add your custom one
2. Create new seed phrase and keypair
3. Import existing seed phrase
4. Deploy SafeMultisig and SURF contract wallets
5. Transfer Crystal and Ruby
6. Create and transfer TIP3 fungible tokens (most advanced version of TIP3 fungible token contracts)
7. Select useful name for public key
8. ton-client-web-js and ton-client-js (partial) injection support
9. Agual.TEAM freeton.domains DeNs resolver
10. Multilanguage (English, Russian, Spanish for now)
11. Darkmode support

Components schema



Backgroundworker

Central expansion module. Provides secure storage of a key pair, secure signature, wallet status control, network status control.

Communicates with the popups and DApps via the RPC interface (ExtensionMessenger)

UI popup

Module user interface. Displays the status of the user's wallets, active actions, allows you to transfer and receive tokens.

UI powered by Framework7 library.

Communicates with the background worker via the RPC interface (ExtensionMessenger)

DApp pages injector

A script that creates methods for getting instances of FreeTON network client libraries (ton-client-web-js and ton-client-web) on each page.

Communicates with the background worker via the RPC interface (TonClientWrapper, NewTonClientWrapper)

Minor modules

Private keys vault

Internal: PrivateStorage.mjs

Stores encrypted keys in browser synchronized storage. Users AES-GCM encryption.

Keyring

Internal: Keyring.mjs

Stores all users public keys and private storage references to keys.

NetworkManager

Internal: NetworkManager.mjs

Provides storage of information about networks, block browsers, as well as the ability to switch the active network

AccountManager

Internal: AccountManager.mjs

Stores and manages user accounts and active wallets.

TokenManager

Internal: TokenManager.mjs

Control and management of user tokens

Extension size and performance

When packed, the extension is approximately 4.5 megabytes in size. Most of this volume is occupied by WASM versions of FreeTON clients used in libraries.

The extension uses the fastest way of communication - the extension message flow. The message flow is wrapped in a special module that implements the asynchronous RPC interface.

System modules: background worker, UI popups, pages with FreeTON clients communicate through the RPM interface.

Installation

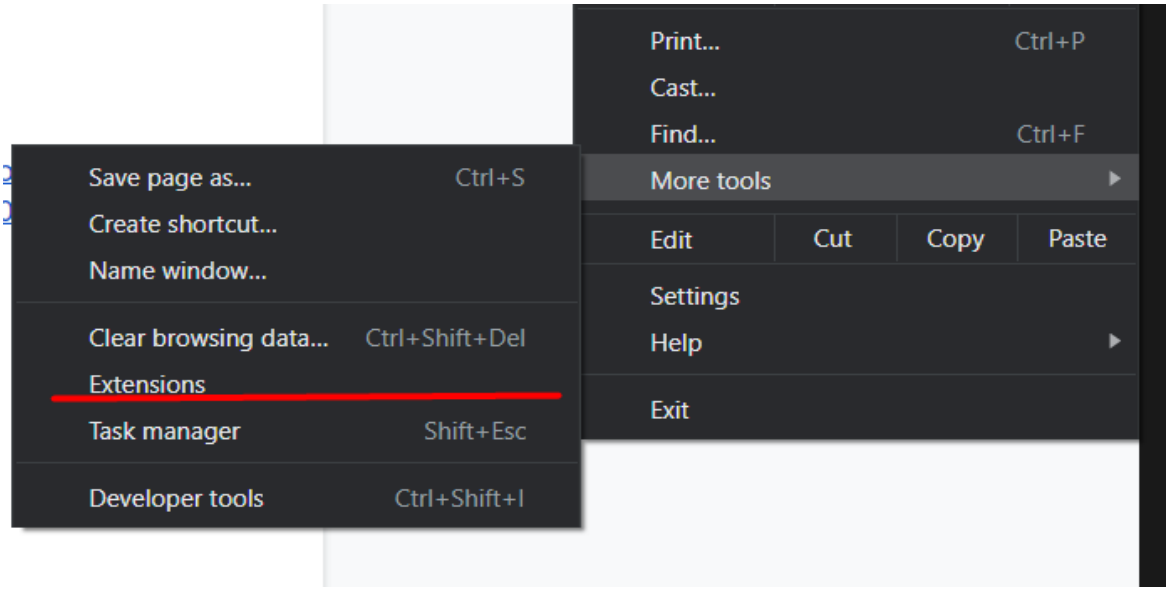
Chromium-based browsers unpacked extension

Tested on Google Chrome, Google Chromium, Microsoft Edge, Opera, Brave, Yandex.Browser and others chromium-based

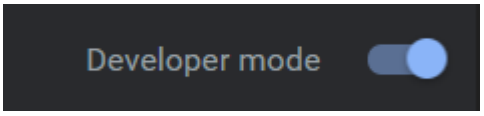
Download all files from main branch of https://github.com/SVOIcom/browser-extension repo or release zip file from https://github.com/SVOIcom/browser-extension/releases/tag/0.0.2

Unpack zip in any place.

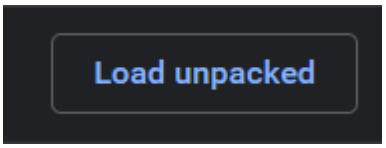
Go to Google Chrome main menu > More tools > Extensions.



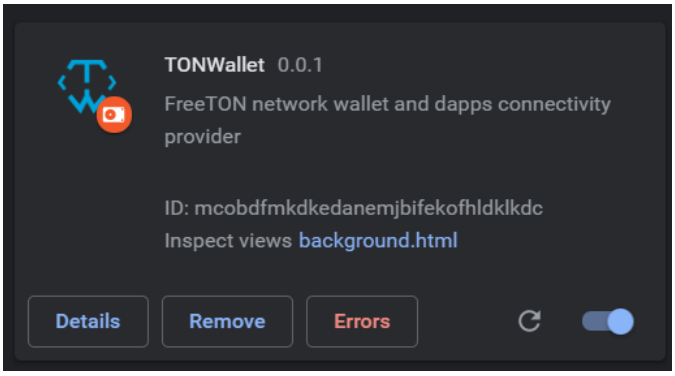
Make sure is developer mode activated



Press “Load unpacked” button and select unpacked extension directory



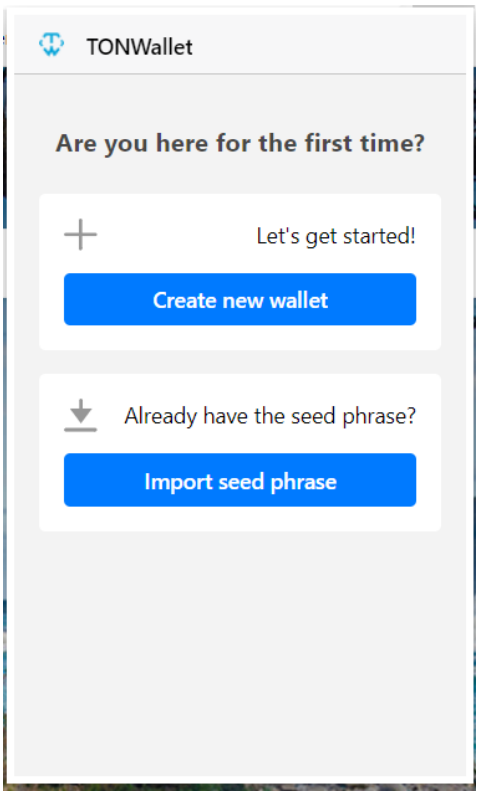
Viola



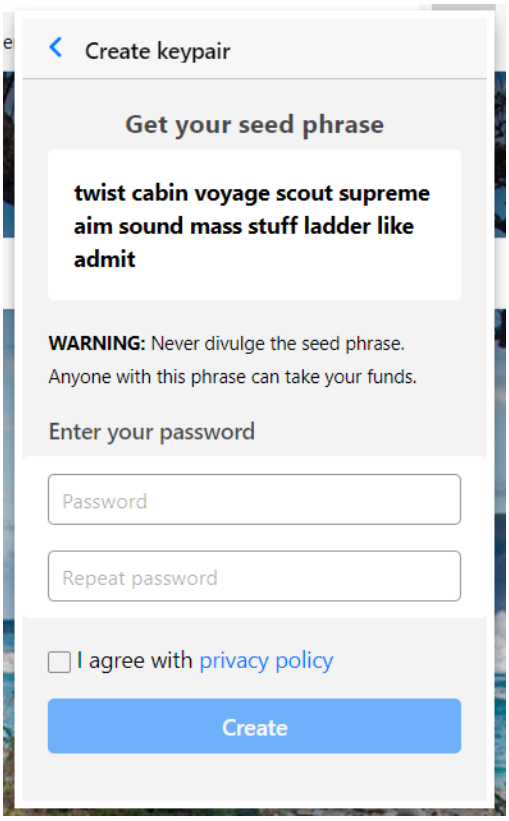
Now you can access the extension from the tiny icon in the top right screen corner.

Usage

In first extension starting you can import your seed phrase or create new one



Let’s create new!



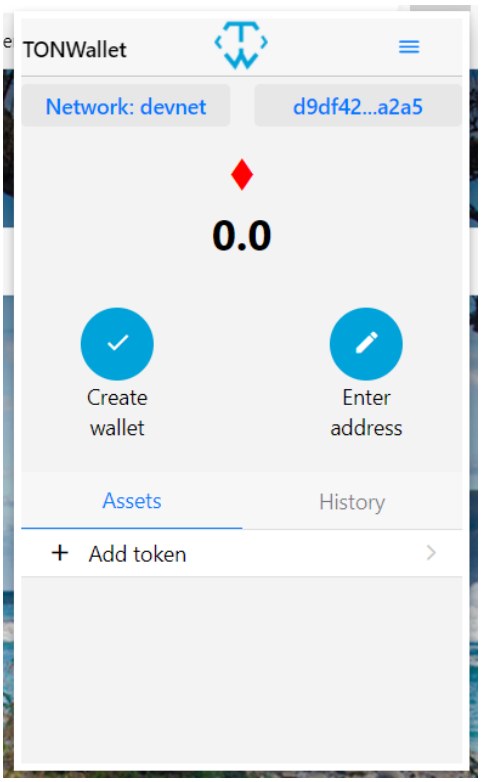
When you open this window, a random seed phrase will be generated, which will NOT be subsequently saved in the extension. Instead, it is immediately converted to a private key, encrypted, and stored into a vault.

Be sure to write down and keep your seed phrase in a reliable source.

Enter the password that this seed phrase will be encrypted. This password will also be required when signing any transactions.

Don’t forget to read Privacy policy ;)

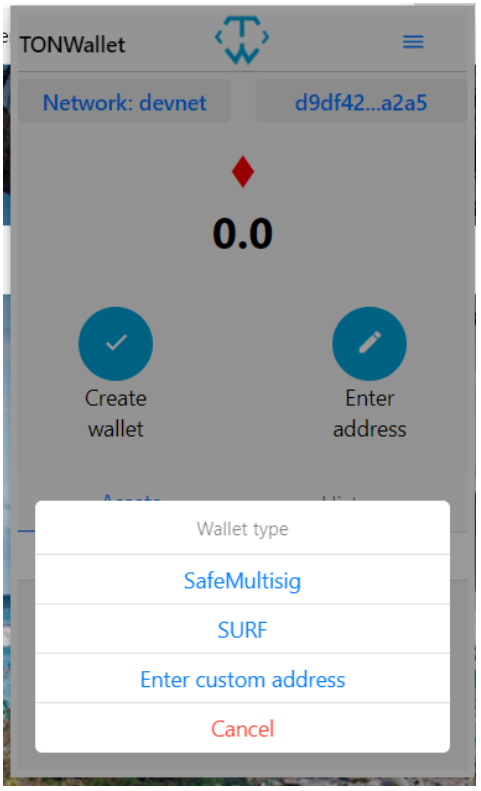
And press create button



Now we are on the main wallet screen.

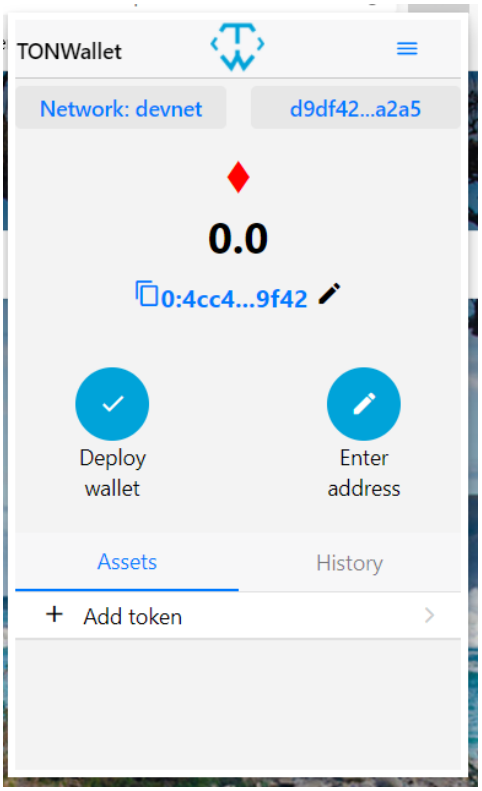
On it you can see the currently selected network, the number of tokens in your wallet, a list of accounts

To start using the wallet, you need to create it in the system. To do this, click on the **Create wallet** button



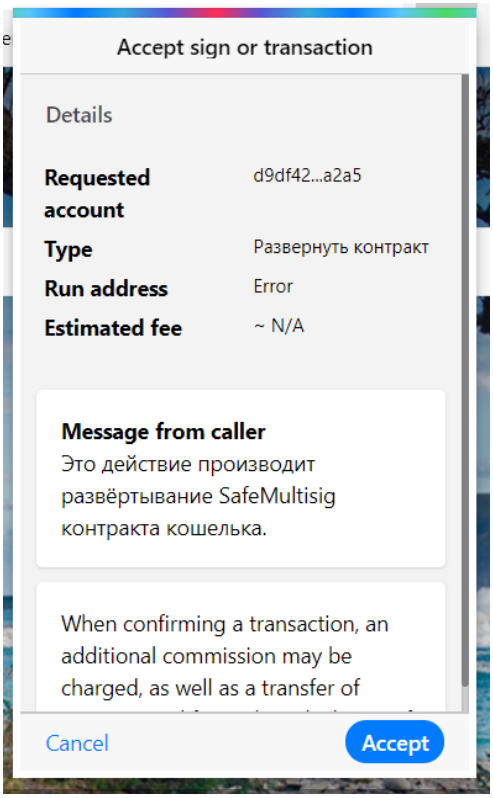
Select the contract you want to use as your main wallet.

The contract can always be changed by clicking on the pencil icon opposite the wallet address.

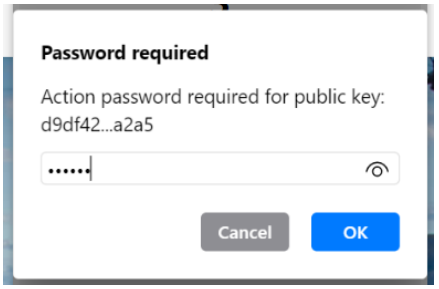


Here is your wallet address. Now let's deploy it.

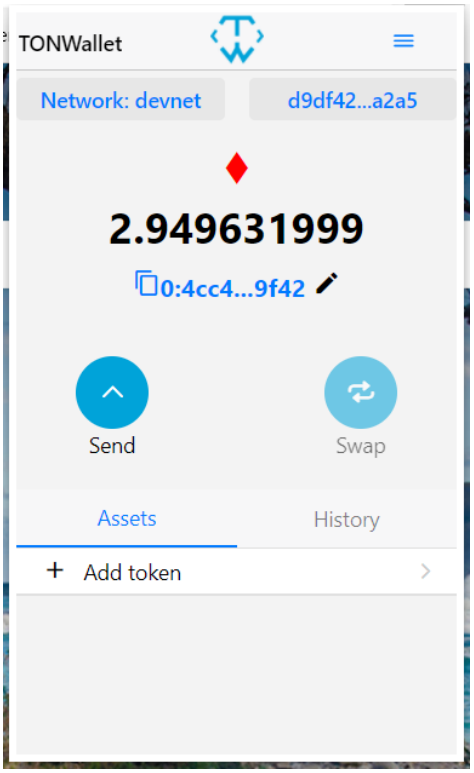
For deployment, you need to get Ruby test tokens or Crystal tokens



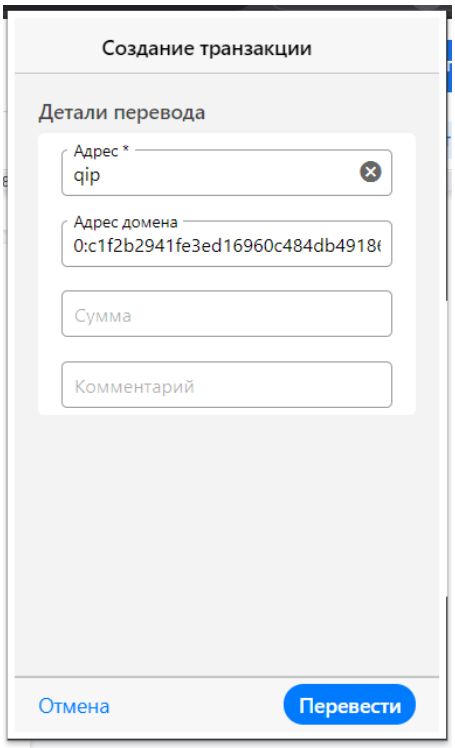
When deploying a wallet contract, you will be prompted to sign the transaction.



To do this, enter the password you specified earlier.



Your wallet is completely ready to go!

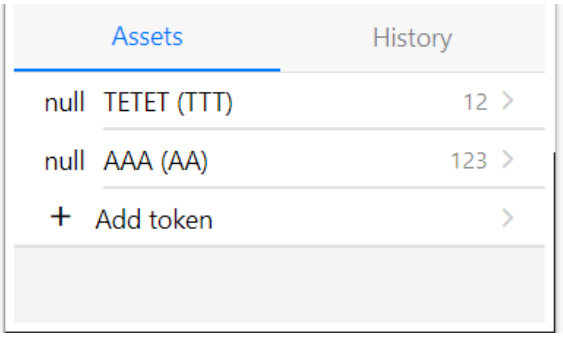


Send money using the transfer window. The address must be the recipient's address or the domain associated with the recipient's address.

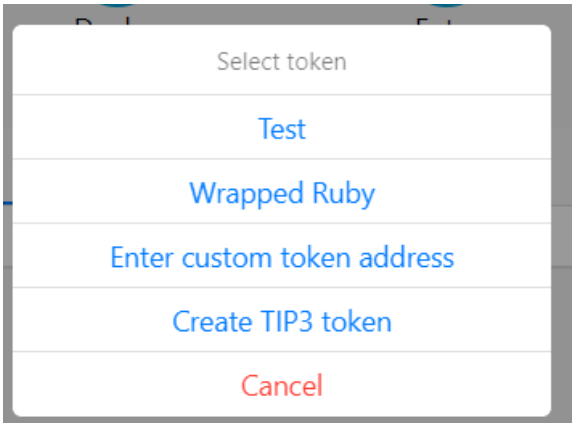
Read more about domains in the Domains section

Tokens

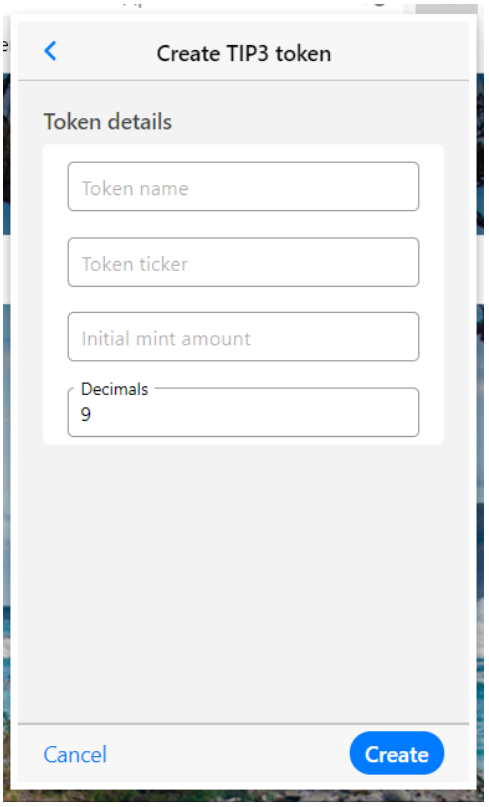
The wallet supports interaction with some types of tokens. (Currently TIP3-fungible contract by Broxus)



You can interact with tokens in the **Assets** section of the wallet

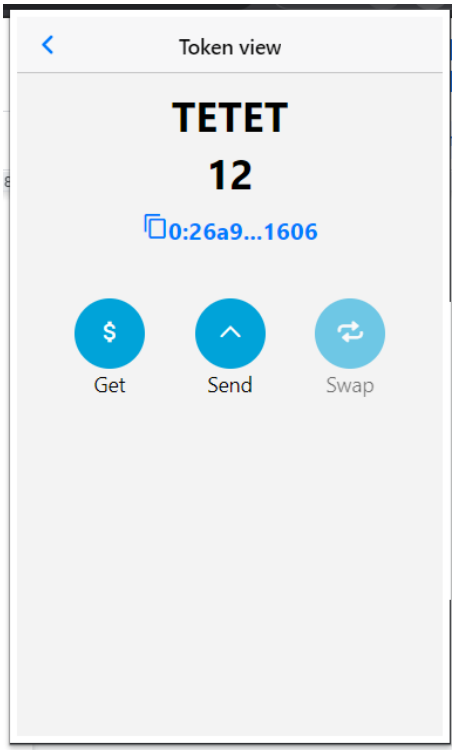


Clicking on the add asset button will prompt you to select one of the existing assets from the list, enter the address of an asset that is not in the list, or open a TIP3 token constructor

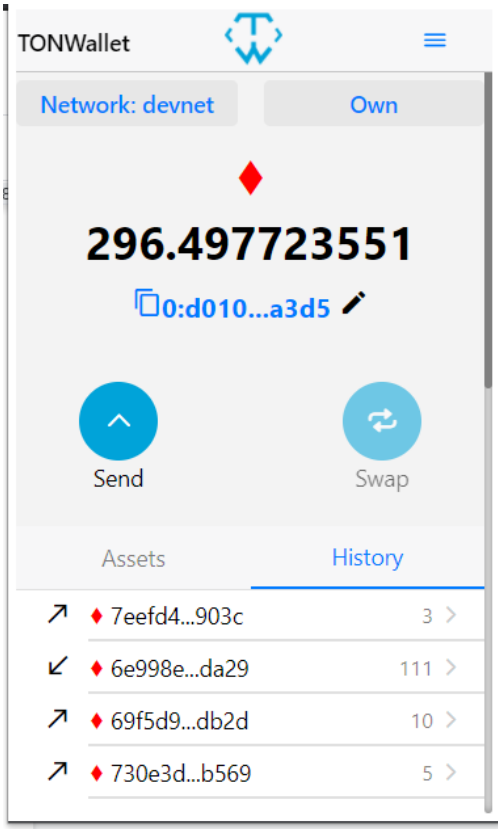


The constructor allows you to specify the desired characteristics of the token, as well as the initial volume of the issue.

After creating a new token, it will be automatically added to your account



The experience of sending and receiving a TIP3 token is similar to sending Crystals or Rubies.



Enjoy the new handy wallet that will make your FreeTON experience enjoyable and convenient

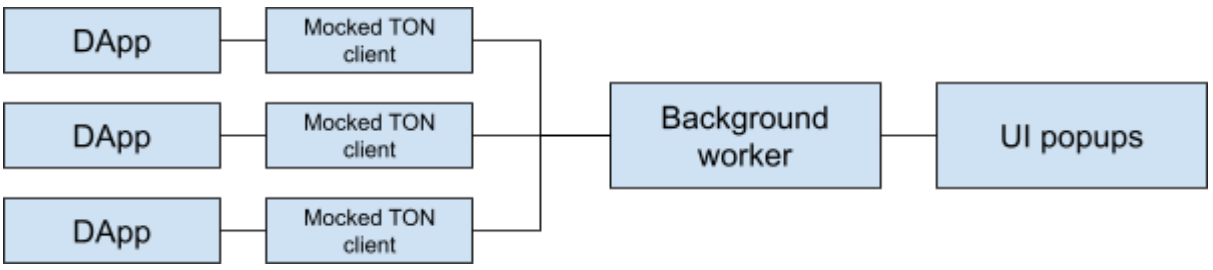
DApp pages API

Our team's goal was to provide developers with the most familiar ways to use the FreeTON network. Therefore, we decided to provide developers with two of the most popular modules for connecting to FreeTON network at once, supplementing them with our modifications and mockes.

TONWallet provides two types of connection modules:

Mocked ton-client-web-js
Partially mocked ton-client-js

Principal scheme



ton-client-web-js - recommended

For full module API see module page https://github.com/tonlabs/ton-client-web-js

Usage:

```
let client = await getTONWeb(); //Returns ton-client-web-js-like instance
```

Mocked methods:

- contracts.run
- contracts.createRunMessage
- contracts.createDeployMessage
- contracts.runLocal

Extra methods:

- async setServers(servers : array) - change current servers

Events:

- NETWORK_CHANGED - networkChanged - emits when network changed
- ACCOUNT_CHANGED - accountChanged - emits when account changed

Events example:

```
let client = await getTONWeb();
client.on(client.EVENTS.NETWORK_CHANGED, (serversUrls)=>{
      console.log('Network changed to',serversUrls)
});
client.on(client.EVENTS.ACCOUNT_CHANGED, (account)=>{
      console.log('Account changed to', account)
});
```

Additional objects:

- accounts
    - async getPublicKeys() - returns all public keys
    - async isKeyInKeyring(publicKey) - checks is public key in keyring
    - async getAccount() - get current account
    - async getWalletInfo() - get current wallet info
    - async getWalletHistory(walletAddress, amount = 20) - get wallet history
    - async getWalletBalance(walletAddress) - get wallet balance
    - async walletTransfer(publicKey, from, to, amount, payload = '') - create transfer from wallet
- extension
    - async getVersion() - returns extension version
- extCrypto
    - async generateSeedPhrase() - generates random seed phrase

    - async getKeysFromSeedPhrase(seedPhrase) - get keypair from seed phrase
- network
    - async get(name = undefined) - returns current network or named
    - async getNetworks() - returns all networks
- utils
    - unsignedNumberToSigned(number, precision=9) - converts unsigned number to signed
    - numberToUnsignedNumber (number, precision=9) - converts signed number to unsigned
    - async asyncWait(milliseconds) - just async wait method
    - shortenPubkey(publicKey) - shorten pubkey for UI
    - hexString2DecString(str) - hex string to number string
    - hexToBase64(str) - converts hex string to base64 encoded string
    - hex2String(string) - converts hex string to regular string
    - string2Hex(str) - converts regular string to hex encoded string
    - validateTONAddress(address) - validates FreeTON address

See API reference for more info

ton-client-js - not fully mocked

For full module API see module page https://github.com/tonlabs/ton-client-js

Usage:

```
let client = await getTONClient(); //Returns ton-client-js-like instance
```

Mocked methods:

No mocked methods for now

Extra methods:

- async setServers(servers) - change current servers

Events:

- NETWORK_CHANGED - networkChanged - emits when network changed
- ACCOUNT_CHANGED - accountChanged - emits when account changed

Additional objects:

- accounts
    - async getPublicKeys() - returns all public keys
    - async isKeyInKeyring(publicKey) - checks is public key in keyring
    - async getAccount() - get current account

    - async getWalletInfo() - get current wallet info
    - async getWalletHistory(walletAddress, amount = 20) - get wallet history
    - async getWalletBalance(walletAddress) - get wallet balance
    - async walletTransfer(publicKey, from, to, amount, payload = '') - create transfer from wallet
- extension
    - async getVersion() - returns extension version
- extCrypto
    - async generateSeedPhrase() - generates random seed phrase
    - async getKeysFromSeedPhrase(seedPhrase) - get keypair from seed phrase
- network
    - async get(name = undefined) - returns current network or named
    - async getNetworks() - returns all networks
- utils
    - unsignedNumberToSigned(number, precision=9) - converts unsigned number to signed
    - numberToUnsignedNumber (number, precision=9) - converts signed number to unsigned
    - async asyncWait(milliseconds) - just async wait method
    - shortenPubkey(publicKey) - shorten pubkey for UI
    - hexString2DecString(str) - hex string to number string
    - hexToBase64(str) - converts hex string to base64 encoded string
    - hex2String(string) - converts hex string to regular string
    - string2Hex(str) - converts regular string to hex encoded string
    - validateTONAddress(address) - validates FreeTON address

See API reference for more info

Code samples

ton-client-web-js simple wallet

index.html file:

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/css/bootstrap.mi
n.css" rel="stylesheet"

integrity="sha384-wEmeIV1mKuiNpC+IOBjI7aAzPcEZeedi5yW5f2yOq55WWLwNGmvvx4
Um1vskeMj0" crossorigin="anonymous">
```

```
    <title>Simple TON wallet</title>
</head>
<body>
<div style="text-align: center">
    <h1 id="balance">0</h1>
    <h2 id="walletAddress">0:0</h2>

    <button id="sendButton">Send TON</button>
</div>


<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/js/bootstrap.bund
le.min.js"

integrity="sha384-p34f1UUtsS3wqzfto5wAAmdvj+osOnFyQFpp4Ua3gs/ZVWx6oOypYo
CJhGGScy+8"
        crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.m
in.js"

integrity="sha384-IQsoLXl5PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ
8NT4GN1R8p"
        crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0/dist/js/bootstrap.min.
js"

integrity="sha384-lpyLfhYuitXl2zRZ5Bn2fqnhNAKOAaM/0Kr9laMspuaMiZfGmfwRNF
h8HlMy49eQ"
        crossorigin="anonymous"></script>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"
integrity="sha256-/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4="
crossorigin="anonymous"></script>
<script src="main.js"></script>
</body>
</html>
```

main.js file:

```
window.addEventListener('load', async () => {

    //Set script start to end of events queue
    setTimeout(async () => {
        if(window.getTONWeb) {
```

```
            //Get TON client instance
            const client = await getTONWeb();
            window.client = client;

            /**
             * Update wallet info method
             * @returns {Promise<void>}
             */
            async function updateWalletInfo() {

                //Get wallet info with address
                let walletInfo = await client.accounts.getWalletInfo();

                //Fetch wallet balance
                let walletBalance = await
client.accounts.getWalletBalance(walletInfo.address);

                //Setup balance field

$('#balance').text(client.utils.unsignedNumberToSigned(walletBalance));

                //Setup wallet address field
                $('#walletAddress').text(walletInfo.address);

            }

            /**
             * Send TONs button handler
             * @returns {Promise<void>}
             */
            async function sendButton() {

                let toAddress = prompt('TON recipient address');

                if(!client.utils.validateTONAddress(toAddress)) {
                    return alert('Invalid TON address')
                }

                let tonsToSend = prompt('How much TONs sending?');

                //Convert to unsigned number
                tonsToSend =
client.utils.numberToUnsignedNumber(tonsToSend);

                if(!tonsToSend) {
                    return alert('Invalid TON amount');
```

```
                }


                //Get wallet info with address
                let walletInfo = await client.accounts.getWalletInfo();

                //Get current user account
                let account = await client.accounts.getAccount();

                //Send TON
                try {
                    await client.accounts.walletTransfer(account.public,
walletInfo.address, toAddress, tonsToSend);
                    alert('TON transferred successfully!')
                } catch (e) {
                    alert('TON sending error: ' + e.message);
                }

            }

            $('#sendButton').click(sendButton);


            //Update wallet info timer
            await updateWalletInfo();
            setInterval(updateWalletInfo, 5000);

            //Update wallet info on network and account changing
            client.on(client.EVENTS.ACCOUNT_CHANGED, updateWalletInfo);
            client.on(client.EVENTS.NETWORK_CHANGED, updateWalletInfo);

        } else {
            alert('No TONWallet extension detected.')
        }
    }, 0);
});
```

result:

DeNs domain

We use DeNs domains API by Augual.TEAM https://github.com/laugual/dens-v2, https://freeton.domains/docs

The wallet makes a request to the api, trying to determine the endpoint of the domain.

If the domain is found and its endpoint is set, then the address of the endpoint is offered for transfer.

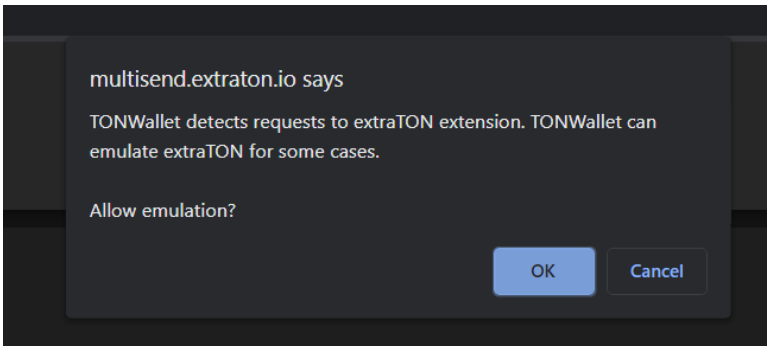
Emulation

For the convenience of using some services, support for the emulation layer of other extensions has been added to our extension. This will allow the user not to install additional extensions to use products working with only one extension.

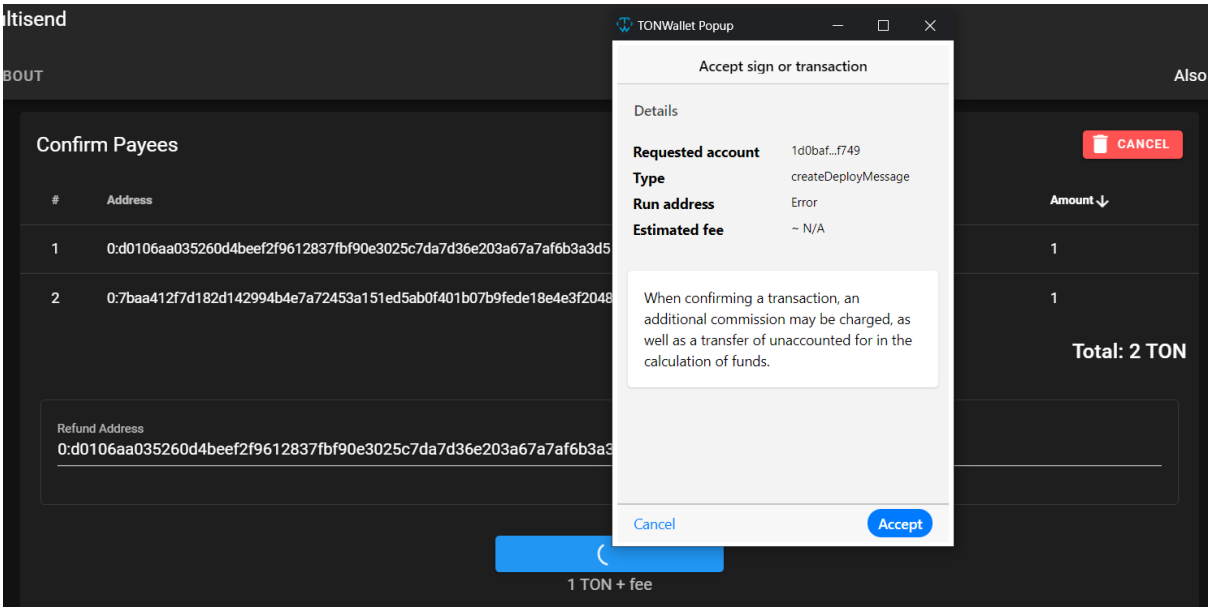
extraTON

One of the added emulation layers emulates the work of some of the extraTON extension methods.

When entering a page designed to use this extension, if it has not been initialized, the user will be prompted to use the emulation mode for the missing extension.



If the answer is OK, the user will be able to sign transactions in TONWallet created by DApp page



API reference

accounts

async getPublicKeys() : [string]

Returns all extension public keys

async isKeyInKeyring(publicKey) : boolean

- **publicKey** - public key for checking

Returns true - if a public key exists in extension keyring. Else - false

async getAccount() : AccountObject

Returns account object:

```
{
 public: "userPublicKey",
 wallets: {
            main: {
                  address: "walletAddress",
                  type: "SafeMultisig"
```

```
            },
             devnet: {
                    address: "walletAddress",
                    type: "SafeMultisig"
            }
  }
 }
```

async getWalletInfo() : WalletObject

Returns current wallet object:

```
{
 address: "walletAddress",
 type: "SafeMultisig"
}
```

async async getWalletHistory(walletAddress, amount = 20) : [HistoryObject]

- **walletAddress** - wallet address
- **amount** - default 20. Amount of history elements for EACH destination. Example: amount 20 returns 40 elements, 20 for outcome and 20 for income

Returns array of HistoryObjects:

```
 [
    {
        boc: "message BOC",
       created_at: "created timestap",
       dst: "destination address",
       id: "message id",
       src: "message source",
       value: 100000 //Message value
    }
 ]
```

async getWalletBalance(walletAddress) : number

- **walletAddress** - wallet address

Returns wallet balance

async walletTransfer(publicKey, from, to, amount, payload = '')

- **publicKey** - user account public key
- **from** - sender wallet address
- **to** - recipient wallet address

- **amount** - sending amount
- **payload** - memo or base64 payload for wallet. Defaults - empty string

Create transfer Crystal or Ruby transaction from user wallet and open sign window

Returns transaction info

extension

async getVersion() : string

Returns extension version string:
`"0.0.2"`

extCrypto

async generateSeedPhrase() : string
Generates random 12-word seed phrase.
Returns string seed:
`"page trim day visual correct rely broom earth style exit drama citizen"`

async getKeysFromSeedPhrase(seedPhrase) : keypairObject

- **seedPhrase** - 12-word seed phrase

returns keyPairObject:

```
{
  public: "47044804595b82538d051080bcd232c40a98c4c7177d357653d43cb613691d53"
  secret: "2a2ece9804e73651ffa36df5907b467c4c01071174da8924498c4616b9f7163a"
}
```

network

async get(name = undefined) : NetworkObject

- name (optional) - network id name. If undefined - returns current network

Returns NetworkObject:

```
{
   name: "main",
       network: {
```

```
            description: "FreeTON main net"
            explorer: "ton.live"
            site: "https://freeton.org/"
            tokenIcon: "💎"
            url: "main.ton.dev"
        }
}
```

async getNetworks() : NetworksObject

Returns object, contains all networks NetworkObject:

```
{
    main: {
            description: "FreeTON main net"
            explorer: "ton.live"
            site: "https://freeton.org/"
            tokenIcon: "💎"
            url: "main.ton.dev"
        }
}
```

utils

validateTONAddress(address) : boolean

- address - TON address

Returns true if address is valid, otherwise - false

unsignedNumberToSigned(number, precision = 9) : number

Converts unsigned number to signed. For example: 1 TON as unsigned is 1000000000

numberToUnsignedNumber (number, precision = 9) : number

Converts regular number to unsigned. For example: 1 TON as unsigned is 1000000000

shortenPubkey(publicKey) : string

Shorten public key for view purposes. Example:

```
let pubkey =
'47044804595b82538d051080bcd232c40a98c4c7177d357653d43cb613691d53';
shortenPubkey(pubkey) //Returns "470448...1d53"
```

Roadmap

We will continue development of the extension and will add a huge amount of new functionality.

1. Implement DeBot browser and DeNs integration with DeBots
2. Implement decentralized ipv4/ipv6 resolving based on DeNs
3. Integrate TonSwap as internal exchange for tokens
4. Integrate nft TIP3 tokens and basic ERC20-like tokken support
5. Direct smart contract interaction by ABI and hash
6. Implement external web-based FreeTON solidity IDE and debugger
7. Integrate more side blockchains
8. Mobile wallet

Contacts

- TG: @svoidev @lailune
- Wallet: 0:0439186aa0147661ebaf2b32ecc76bac172fcdaa24c7df7c9cb03cc816e435e6