Wallet Integrations

Learn how to integrate various web3 wallets with SecretJS.

Secret Network Client Setup

```

Copy import{ SecretNetworkClient,Wallet }from"secretjs";

constwallet=newWallet("Your mnemonic words go here");

constsecretjs=newSecretNetworkClient({ chainId:"pulsar-3", url:"https://api.pulsar3.scrttestnet.com", wallet:wallet, walletAddress:wallet.address, });

```

Metamask

```

Copy import{ SecretNetworkClient,MetaMaskWallet }from"secretjs";

//@ts-ignore const[ethAddress]=awaitwindow.ethereum.request({ method:"eth_requestAccounts", });

constwallet=awaitMetaMaskWallet.create(window.ethereum,ethAddress);

constsecretjs=newSecretNetworkClient({ url:"TODO get from https://github.com/scrtlabs/api-registry", chainId:"secret-4", wallet:wallet, walletAddress:wallet.address, });

```

Notes:

1. MetaMask supports mobile!
2. MetaMask supports Ledger.
3. SecretjsMetaMaskWallet
4. will automatically prompt the user to sign apersonal_sign
5. message, which is used to recover the users public key and derive the user's Secret Network address.
6. You might want to passencryptionSeed
7. toSecretNetworkClient.create()
8. to use the same encryption key for the user across sessions. This value should be a true random 32 byte number that is stored securly in your app, such that only the user can decrypt it. This can also be asha256(user_password)
9. but might impair UX.
10. See Keplr'sgetOfflineSignerOnlyAmino()
11. for list of unsupported transactions.
12.

Keplr Walllet

The recommended way of integrating Keplr is by usingwindow.keplr.getOfflineSignerOnlyAmino() :

```

Copy import{ SecretNetworkClient }from"secretjs";

constsleep=(ms:number)=>newPromise((resolve)=>setTimeout(resolve,ms));

while( !window.keplr|| !window.getEnigmaUtils|| !window.getOfflineSignerOnlyAmino ) { awaitsleep(50); }

constCHAIN_ID="secret-4";

awaitwindow.keplr.enable(CHAIN_ID);

constkeplrOfflineSigner=window.keplr.getOfflineSignerOnlyAmino(CHAIN_ID); const[{ address:myAddress}]=awaitkeplrOfflineSigner.getAccounts();

consturl="TODO get from https://github.com/scrtlabs/api-registry";

constsecretjs=newSecretNetworkClient({ url, chainId:CHAIN_ID, wallet:keplrOfflineSigner, walletAddress:myAddress, encryptionUtils:window.keplr.getEnigmaUtils(CHAIN_ID), });
```

// Note: Using window.getEnigmaUtils is optional, it will allow // Keplr to use the same encryption seed across sessions for the account. // The benefit of this is that secretjs.query.getTx() will be able to decrypt // the response across sessions.

```

Notes:

1. No mobile support yet.
2. Keplr supports Ledger.
3. By usingencryptionUtils
4. you let Keplr handle user encryption keys for you, which allows you to easily decrypt transactions across sessions.
5.

Links:

- Official Keplr Website »
- Keplr API Docs »
-

SignerOnlyAmino vsSigner vsSignerAuto

TLDR:

- getOfflineSignerOnlyAmino()
- : The recommended way. Supports Ledger, has a nice UI.
- getOfflineSigner()
- : No Ledger support, ugly UI, can send IBCrelayer
- txs and submit IBC gov proposals.
- getOfflineSignerAuto()
- : If Ledger alias forgetOfflineSignerOnlyAmino()
- , otherwise alias forgetOfflineSigner()
- .
-

window.keplr.getOfflineSignerOnlyAmino()

Although this is the legacy way of signing transactions on cosmos-sdk, it's still the most recommended for connecting to Keplr due to Ledger support & better UI on Keplr.

- Looks good on Keplr
- Supports users signing with Ledger
- Doesn't support signing these transactions:
-
  - Every tx type underibc_client
  - ,ibc_connection
  - andibc_channel
  - (meaning IBC relaying, for example withts-relayer
  - )
  - gov/MsgSubmitProposal/ClientUpdateProposal
  - gov/MsgSubmitProposal/UpgradeProposal
- *
-

Note thatibc_transfer/MsgTransfer for sending funds across IBCis supported.

window.keplr.getOfflineSigner()

The new way of signing transactions on cosmos-sdk, it's more efficient but still doesn't have Ledger support, so it's most recommended for usage in apps that don't require signing transactions with Ledger.

- Looks bad on Keplr
- Doesn't support users signing with Ledger
- Supports signing transactions with all types of Msgs
-
```

window.keplr.getOfflineSignerAuto()

If the connected Keplr account uses Ledger, returnswindow.keplr.getOfflineSignerOnlyAmino() . Otherwise returnswindow.keplr.getOfflineSigner() .

Fina Wallet

Fina implements the Keplr API, sothe above Keplr docs applies. If you support Keplr, your app will also work on the Fina Wallet mobile app. This works because the Fina Wallet mobile app has webview to which it injects its objects underwindow.keplr .

Fina supports deep linking into its in-app browser.

Example1:fina://wllet/dapps?network=secret-4&url=https%3A%2F%2Fdash.scrt.network

Example2:

If a user accessed your app using a regular mobile browser, you can open your app in the Fina in-app browser using this code:

```

Copy consturlSearchParams=newURLSearchParams(); urlSearchParams.append("network","secret-4"); urlSearchParams.append("url",window.location.href);

window.open(fina://wllet/dapps?{urlSearchParams.toString()},"_blank");

```

Links:

- Official Fina Website »
-

Leap Cosmos Wallet

The recommended way of integrating Leap is by usingwindow.leap.getOfflineSignerOnlyAmino() :

```

Copy import{ SecretNetworkClient }from"secretjs";

constsleep=(ms:number)=>newPromise((resolve)=>setTimeout(resolve,ms));

while( !window.leap|| !window.leap.getEnigmaUtils|| !window.leap.getOfflineSignerOnlyAmino ) { awaitsleep(50); }

constCHAIN_ID="secret-4";

awaitwindow.leap.enable(CHAIN_ID);

constleapOfflineSigner=window.leap.getOfflineSignerOnlyAmino(CHAIN_ID); const[{ address:myAddress}]=awaitleapOfflineSigner.getAccounts();

consturl="TODO get from https://github.com/scrtlabs/api-registry";

constsecretjs=newSecretNetworkClient({ url, chainId:CHAIN_ID, wallet:leapOfflineSigner, walletAddress:myAddress, encryptionUtils:window.leap.getEnigmaUtils(CHAIN_ID), });

// Note: Using window.leap.getEnigmaUtils() is optional, it will allow // Leap to use the same encryption seed across sessions for the account. // The benefit of this is that secretjs.query.getTx() will be able to decrypt // the response across sessions.

```

Links:

- Official Leap Website »
- Leap API Docs »
-

Starshell Wallet

StarShell implements the Keplr API, sothe above Keplr docs applies. If you support Keplr, your app will also work on StarShell wallet. This works because StarShell wallet asks the user to turn off Keplr and then overrideswindow.keplr with its

objects.

Links:

- Official StarShell Website »
-

Ledger Wallet

@cosmjs/ledger-amino can be used to sign transactions with a Ledger wallet running the Cosmos app.

```

Copy import{ SecretNetworkClient }from'secretjs'; import{ makeCosmoshubPath }from"@cosmjs/amino"; import{ LedgerSigner }from"@cosmjs/ledger-amino";

// NodeJS only importTransportNodeHidfrom"@ledgerhq/hw-transport-node-hid";

// Browser only //import TransportNodeHid from "@ledgerhq/hw-transport-webusb";

constinteractiveTimeout=120_000; constaccountIndex=0; constcosmosPath=makeCosmoshubPath(accountIndex);

constledgerTransport=awaitTransportNodeHid.create(interactiveTimeout,interactiveTimeout);
constledgerSigner=newLedgerSigner( ledgerTransport, { testModeAllowed:true, hdPaths:[cosmosPath], prefix:'secret' } );
const[{address}]=awaitsigner.getAccounts();

constclient=newSecretNetworkClient({ url:"TODO get from https://github.com/scrtlabs/api-registry", chainId:"secret-4", wallet:ledgerSigner, walletAddress:address, });

```

Notes:

1. Use the appropriatehw-transport
2. package for your environment (Node or Browser)
3. The Ledger Cosmos app only supports coin type 118
4. You might want to passencryptionSeed
5. toSecretNetworkClient.create()
6. to use the same encryption key for the user across sessions. This value should be a true random 32 byte number that is stored securly in your app, such that only the user can decrypt it. This can also be asha256(user_password)
7. but might impair UX.
8. See Keplr'sgetOfflineSignerOnlyAmino()
9. for list of unsupported transactions.
10.

Links:

- @cosmjs/ledger-amino Documentation
-

Last updated2 months ago On this page *Secret Network Client Setup * Metamask * Keplr Walllet * Fina Wallet * Leap Cosmos Wallet * Starshell Wallet * Ledger Wallet

Was this helpful? Edit on GitHub Export as PDF