LoginID Fido Vault SDK

Fido Vault SDK is a javascript library to securely connect and sign transaction with Fido Vault.

Add SDK to Application

Install using npm:

```
npm install @loginid/fidovault-sdk
```

Install using CDN:

```
<script src="https://loginid-sdk.s3-us-west-1.amazonaws.com/sdk/js/0.9.8/loginid.fidovault-sdk.min.js">
```

Create SDK Instance

```
import {FidoVaultSDK} from "@loginid/fidovault-sdk";

// initialize wallet instance
const wallet = new FidoVaultSDK(process.env.VAULT_URL || "https://vault-qa.awstest.logi
```

Fido Vault Addresses Discovery API

An API for a function used to discover the addresses a wallet user is willing to use for a given Algorand DApp.

Function:

```
async enable(network: EnableOpts): Promise<EnableResult>
```

Interfaces:

```
// support optional network "mainnet | testnet | sandnet"
export interface EnableOpts {
    network?: string;
    genesisID?: string;
    genesisHash?: string;
}

export interface EnableResult {
    genesisID: string;
    genesisHash: string;
    accounts: AlgorandAddress[];
}
```

Example:

```
    try {
        const result = await wallet.enable({ network: "sandnet" });
        if (result != null) {
            // store user addresses
            localStorage.setItem("addresses", result.accounts);
        }
    } catch (error) {
        console.log(error);
    }
```

Fido Vault Transaction Signing API

An API for a function used to sign a list of transactions on the Algorand blockchain.

Function:

```
async signTxns(txns: WalletTransaction[], opts?: SignTxnsOpts): Promise<TxnsResult>
```

Interfaces:

```
export interface TxnsResult {
    txnIds: TxnId[];
    signTxn: string[];
}

export interface WalletTransaction {
    /**
     * Base64 encoding of the canonical msgpack encoding of a Transaction.
     */
    txn: string;
    /**
    * Optional authorized address used to sign the transaction when the account
    * is rekeyed. Also called the signor/sgnr.
    */
    authAddr?: AlgorandAddress;

    /**
     * Optional list of addresses that must sign the transactions
     */
    signers?: AlgorandAddress[];

    /**
     * Optional base64 encoding of the canonical msgpack encoding of a
     * SignedTxn corresponding to txn, when signers=[]
     */
    stxn?: string;

    /**
     * Optional message explaining the reason of the transaction
     */
    message?: string;

    /**
     * Optional message explaining the reason of this group of transaction
     * Field only allowed in the first transaction of a group
     */
    groupMessage?: string;
}
```

Example Sign Transaction:

```
try {
    // construct a transaction note
    const note = new Uint8Array(Buffer.from("Simple Payment", "utf8"));
    // get user address from discovery api
    const addr = localStorage.getItem("user_default_address");
    // dapp recieving address
    const receiver =
        process.env.REACT_APP_DAPP_ADDRESS ||
        "OZL4D23EET2S44UJBHZGHSMUQPJSA5YK7X4J737N5QZUJY3WE4X6PFHIXE";

    // create a payment transaction using algosdk from official algorand js sdk (https:/
    const txn = algosdk.makePaymentTxnWithSuggestedParamsFromObject({
        from: addr,
        to: receiver,
        amount: 1000000,
        note,
        suggestedParams: params,
    });

    // initialize WalletTransaction interface
    let wTxn: WalletTransaction = {
        txn: Buffer.from(txn.toByte()).toString("base64"),
        signers: [addr],
    };
    // request signing from Fido Vault
    const res = await wallet.signTxns([wTxn]);

    // send sign transaction to algorand node
    const post = await postTransaction(res.signTxn);
    console.log(post);
} catch (error) {
    console.log(error);
}
```