<repository>

LUKSO-Hardhat-template

overview

repository: https://github.com/CJ42/LUKSO-Hardhat-template/
userName: CJ42
repository: LUKSO-Hardhat-template
branch: main
date: 2023-11-27T10:50:57+01:00 (1701078657)

<file>

path: /.env.example

url: https://github.com/CJ42/LUKSO-Hardhat-template/blob/main/.env.example

```
PRIVATE_KEY=
```

</file>

<file>

path: /.nvmrc

url: https://github.com/CJ42/LUKSO-Hardhat-template/blob/main/.nvmrc

```
16.20
```

</file>

<file>

path: /.prettierrc

url: https://github.com/CJ42/LUKSO-Hardhat-template/blob/main/.prettierrc

```
{
  "plugins": ["prettier-plugin-solidity"],
  "overrides": [
    {
      "files": ["*.js", "*.ts"],
      "options": {
        "tabWidth": 2,
        "printWidth": 100,
        "trailingComma": "all",
        "singleQuote": true,
        "semi": true
      }
    },
    {
      "files": "*.sol",
      "options": {
        "tabWidth": 4,
        "printWidth": 80,
        "compiler": "0.8.15"
      }
    }
  ]
}
```

</file>

<file>

path: /.tool-versions

url: https://github.com/CJ42/LUKSO-Hardhat-template/blob/main/.tool-versions

```
nodejs 16.20.2
```

</file>

<file>

path: /README.md

url: https://github.com/CJ42/LUKSO-Hardhat-template/blob/main/README.md

LUKSO Hardhat Template

This project demonstrates how to use LSP7 or LSP8 to create a basic NFT collection on LUKSO using the [@lukso/lsp-smart-contracts (https://github.com/lukso-network/universalprofile-smart-contracts)](https://github.com/lukso-network/universalprofile-smart-contracts);

It uses:

- Hardhat v2.18.2
- Ethers v6

- Artifacts generated by Typechain

Setup

Make sure you are using node js v16.20. You can set this up with nvm (https://github.com/nvm-sh/nvm).

```
nvm install 16.20
nvm use 16.20
```

Installation

```
git clone https://github.com/CJ42/LUKSO-Hardhat-template.git
cd LUKSO-Hardhat-Template
yarn
```

</file>

<file>

path: /contracts/Example 1/EventTicketsNFT.sol

url: https://github.com/CJ42/LUKSO-Hardhat-template/blob/main/contracts/Example 1/EventTicketsNFT.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

// modules
import {
    LSP7Mintable
} from "@lukso/lsp-smart-contracts/contracts/LSP7DigitalAsset/presets/LSP7Mintable.sol";

import {_LSP4_TOKEN_TYPE_DATA_KEY, TokenType} from "../TokenTypes.sol";

contract EventTicketsNFT is LSP7Mintable {
    constructor(
        string memory eventName,
        string memory tokenSymbol,
        address contractOwner
    )
        LSP7Mintable(
            eventName,
            tokenSymbol,
            contractOwner,
            true // make the token non divisible
        )
    {
        // set the token type
        _setData(_LSP4_TOKEN_TYPE_DATA_KEY, abi.encode(TokenType.TOKEN));
    }
}
```

</file>

<file>

path: /contracts/Example 2/DigitalTradingCards.sol

url: https://github.com/CJ42/LUKSO-Hardhat-template/blob/main/contracts/Example 2/DigitalTradingCards.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

// modules
import {
    LSP7Mintable
} from "@lukso/lsp-smart-contracts/contracts/LSP7DigitalAsset/presets/LSP7Mintable.sol";

import {_LSP4_TOKEN_TYPE_DATA_KEY, TokenType} from "../TokenTypes.sol";

contract DigitalTradingCards is
    LSP7Mintable(
        "Digital Trading Card",
        "DTC",
        msg.sender,
        true // cards are non divisible (you cannot tear a card in 2 and give me half)
    )
{
    constructor() {
        // set the token type
        _setData(_LSP4_TOKEN_TYPE_DATA_KEY, abi.encode(TokenType.NFT));
    }
}
```

</file>

<file>

path: /contracts/Example 3/BasicNFTCollection.sol

url: https://github.com/CJ42/LUKSO-Hardhat-template/blob/main/contracts/Example 3/BasicNFTCollection.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

// modules
import {
    LSP8Mintable
} from "@lukso/lsp-smart-contracts/contracts/LSP8IdentifiableDigitalAsset/presets/LSP8Mintable.sol";

// constants
import {
    _LSP8_TOKENID_TYPE_NUMBER
} from "@lukso/lsp-smart-contracts/contracts/LSP8IdentifiableDigitalAsset/LSP8Constants.sol";
import {_LSP4_TOKEN_TYPE_DATA_KEY, TokenType} from "../TokenTypes.sol";

contract BasicNFTCollection is LSP8Mintable {
    constructor(
        string memory nftCollectionName,
        string memory nftCollectionSymbol,
        address contractOwner
    )
        LSP8Mintable(
            nftCollectionName,
            nftCollectionSymbol,
            contractOwner,
            _LSP8_TOKENID_TYPE_NUMBER
        )
    {
        // set token type
        _setData(_LSP4_TOKEN_TYPE_DATA_KEY, abi.encode(TokenType.COLLECTION));
    }
}
```

</file>

<file>

path: /contracts/Example 4/DynamicNFTCollection.sol

url: https://github.com/CJ42/LUKSO-Hardhat-template/blob/main/contracts/Example 4/DynamicNFTCollection.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

// modules
import {
    LSP8Mintable
} from "@lukso/lsp-smart-contracts/contracts/LSP8IdentifiableDigitalAsset/presets/LSP8Mintable.sol";

// constants
import {
    _LSP8_TOKENID_TYPE_ADDRESS
} from "@lukso/lsp-smart-contracts/contracts/LSP8IdentifiableDigitalAsset/LSP8Constants.sol";

import {_LSP4_TOKEN_TYPE_DATA_KEY, TokenType} from "../TokenTypes.sol";

contract DynamicNFTCollection is LSP8Mintable {
    constructor(
        string memory dynamicNFTCollectionName,
        string memory dynamicNFTCollectionSymbol,
        address contractOwner
    )
        LSP8Mintable(
            dynamicNFTCollectionName,
            dynamicNFTCollectionSymbol,
            contractOwner,
            _LSP8_TOKENID_TYPE_ADDRESS
        )
    {
        // Set the type of the token
        _setData(_LSP4_TOKEN_TYPE_DATA_KEY, abi.encode(TokenType.COLLECTION));
    }
}
```

</file>

<file>

path: /contracts/Example 4/DynamicNFTMetadataContract.sol

url: https://github.com/CJ42/LUKSO-Hardhat-template/blob/main/contracts/Example 4/DynamicNFTMetadataContract.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

// modules
import {ERC725Y} from "@erc725/smart-contracts/contracts/ERC725Y.sol";

contract DynamicNFT is ERC725Y {

    bytes32 constant _LSP8_REFERENCE_CONTRACT =
0x708e7b881795f2e6b6c2752108c177ec89248458de3bf69d0d43480b3e5034e6;

    constructor(address nftOwner, address nftCollection) ERC725Y(nftOwner) {
        /**
         * @dev set the reference to the NFT collection that this metadata contract belongs to
         *
         * {
         *     "name": "LSP8ReferenceContract",
         *     "key": "0x708e7b881795f2e6b6c2752108c177ec89248458de3bf69d0d43480b3e5034e6",
         *     "keyType": "Singleton",
         *     "valueType": "(address,bytes32)",
         *     "valueContent": "(Address,bytes32)"
         * }
         */
        _setData(_LSP8_REFERENCE_CONTRACT, abi.encodePacked(nftCollection,
bytes32(bytes20(address(this)))));
    }
}
```

</file>

<file>

path: /contracts/Example 5/LSP7DigitalStickers.sol

url: https://github.com/CJ42/LUKSO-Hardhat-template/blob/main/contracts/Example 5/LSP7DigitalStickers.sol

```sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

import {
    LSP7DigitalAsset
} from "@lukso/lsp-smart-contracts/contracts/LSP7DigitalAsset/LSP7DigitalAsset.sol";
import {
    LSP7Burnable
} from "@lukso/lsp-smart-contracts/contracts/LSP7DigitalAsset/extensions/LSP7Burnable.sol";

import {_LSP4_TOKEN_TYPE_DATA_KEY, TokenType} from "../TokenTypes.sol";

contract LSP7DigitalStickers is LSP7DigitalAsset, LSP7Burnable {
    constructor(
        string memory stickerName,
        string memory stickerSymbol,
        address contractOwner,
        uint256 totalNumberOfStickersInCirculation
    ) LSP7DigitalAsset(stickerName, stickerSymbol, contractOwner, true) {
        // Set the type of the token
        _setData(_LSP4_TOKEN_TYPE_DATA_KEY, abi.encode(TokenType.NFT));

        // mint the total number of stickers that the LSP8 Collection owns
        _mint({
            to: msg.sender,
            amount: totalNumberOfStickersInCirculation,
            force: true,
            data: abi.encodePacked(
                "creating ",
                totalNumberOfStickersInCirculation,
                " new stickers for this collection!"
            )
        });
    }
}
```

</file>

<file>

path: /contracts/Example 5/LSP8DigitalStickersCollection.sol

url: https://github.com/CJ42/LUKSO-Hardhat-template/blob/main/contracts/Example 5/LSP8DigitalStickersCollection.sol

```sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

// modules
import {
    LSP8IdentifiableDigitalAsset
```

```
} from "@lukso/lsp-smart-contracts/contracts/LSP8IdentifiableDigitalAsset/LSP8IdentifiableDigitalAsset.sol";
import {LSP7DigitalStickers} from "./LSP7DigitalStickers.sol";

// constants
import {
    _LSP8_TOKENID_TYPE_ADDRESS
} from "@lukso/lsp-smart-contracts/contracts/LSP8IdentifiableDigitalAsset/LSP8Constants.sol";

import {_LSP4_TOKEN_TYPE_DATA_KEY, TokenType} from "../TokenTypes.sol";

/**
 * @dev Pseudo code example, LSP8 collection of digital stickers, where each sticker is a LSP7 contract
with a limited supply available.
 * This contract is for testing and tutorial purpose.
 */
contract LSP8DigitalStickersCollection is
    LSP8IdentifiableDigitalAsset(
        "Example Digital Stickers Collection",
        "EDSC",
        msg.sender,
        _LSP8_TOKENID_TYPE_ADDRESS
    )
{
    constructor() {
        // Set the type of the token
        _setData(_LSP4_TOKEN_TYPE_DATA_KEY, abi.encode(TokenType.COLLECTION));
    }

    // Constants ---

    bytes32 internal constant _LSP8_REFERENCE_CONTRACT_DATA_KEY =
        0x708e7b881795f2e6b6c2752108c177ec89248458de3bf69d0d43480b3e5034e6;

    bytes32 internal constant _STICKER_PRICE_DATA_KEY =
        keccak256("Price Per Sticker");

    // Errors ---

    error FailedDeployingLSP7StickerContract(string stickerName);

    function createNewStickerCollection(
        string calldata stickerName,
        string calldata stickerSymbol,
        uint256 numberOfStickers,
        uint256 pricePerSticker // in wei
    ) external onlyOwner {
        try
            new LSP7DigitalStickers(
                stickerName,
                stickerSymbol,
                address(this),
                numberOfStickers
```

```
            )
        returns (LSP7DigitalStickers newSticker) {
            // example:
            // newSticker (contract address)= 0x9760e8347D5E8f9042726E3B4DdE9Ae787476101

            // NFT tokenId will be
0x0000000000000000000000009760e8347D5E8f9042726E3B4DdE9Ae787476101
            bytes32 stickerId = bytes32(abi.encode(address(newSticker)));

            bytes32[] memory dataKeysToSet = new bytes32[](2);
            bytes[] memory dataValuesToSet = new bytes[](2);

            // Set Data Key "LSP8ReferenceContract",
            dataKeysToSet[0] = _LSP8_REFERENCE_CONTRACT_DATA_KEY;
            dataValuesToSet[0] = abi.encodePacked(address(this), stickerId); // "valueType": "
(address,bytes32)"

            // Set Data Key specific to the price per sticker
            dataKeysToSet[1] = _STICKER_PRICE_DATA_KEY;
            dataValuesToSet[1] = abi.encode(pricePerSticker);

            newSticker.setDataBatch(dataKeysToSet, dataValuesToSet);

            // need to pass  force = true  as will attempt to notify  universalReceiver 
function of this contract
            // which does not implement LSP1
            _mint({
                to: address(this),
                tokenId: stickerId,
                force: true,
                data: ""
            });
        } catch (bytes memory) {
            revert FailedDeployingLSP7StickerContract(stickerName);
        }
    }
}
```

</file>

<file>

path: /contracts/Playground.sol

url: https://github.com/CJ42/LUKSO-Hardhat-template/blob/main/contracts/Playground.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

// modules
import {
    LSP8Mintable
} from "@lukso/lsp-smart-contracts/contracts/LSP8IdentifiableDigitalAsset/presets/LSP8Mintable.sol";

// constants
import {
    _LSP8_TOKENID_TYPE_STRING
} from "@lukso/lsp-smart-contracts/contracts/LSP8IdentifiableDigitalAsset/LSP8Constants.sol";

// put the address to specify the owner of the LSP8 collection and use a different deployer address
address constant _LSP8_COLLECTION_OWNER = 0xCAfEcAfeCAfECaFeCaFecaFecaFECafECafeCaFe;

contract DigitalArtCollection is
    LSP8Mintable(
        "3D Artistic Cubes", // collection name
        "3DCuB", // symbol
        _LSP8_COLLECTION_OWNER, // contract owner
        _LSP8_TOKENID_TYPE_STRING
    )
{
    function createNewCube(
        string memory cubeName,
        address recipient
    ) public onlyOwner {
        bytes32 tokenId = bytes32(bytes(cubeName));
        _mint({to: recipient, tokenId: tokenId, force: true, data: ""});
    }
}
```

</file>

<file>

path: /contracts/TokenTypes.sol

url: https://github.com/CJ42/LUKSO-Hardhat-template/blob/main/contracts/TokenTypes.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

bytes32 constant _LSP4_TOKEN_TYPE_DATA_KEY =
0xe0261fa95db2eb3b5439bd033cda66d56b96f92f243a8228fd87550ed7bdfdb3;

enum TokenType {
    TOKEN,
    NFT,
    COLLECTION
}
```

</file>

<file>

path: /hardhat.config.ts

url: https://github.com/CJ42/LUKSO-Hardhat-template/blob/main/hardhat.config.ts

```
import { HardhatUserConfig } from 'hardhat/config';
import { NetworkUserConfig } from 'hardhat/types';
import '@nomicfoundation/hardhat-toolbox';

import * as dotenv from 'dotenv';
dotenv.config();

function getTestnetChainConfig(): NetworkUserConfig {
  const config: NetworkUserConfig = {
    url: 'https://rpc.testnet.lukso.network',
    chainId: 4201,
  };

  if (process.env.PRIVATE_KEY !== undefined) {
    config['accounts'] = [process.env.PRIVATE_KEY];
  }

  return config;
}

const config: HardhatUserConfig = {
  solidity: {
    version: '0.8.19',
    settings: {
      optimizer: {
        enabled: true,
        runs: 200,
      },
    },
  },
  typechain: {
    outDir: 'typechain-types',
    target: 'ethers-v6',
  },
  networks: {
    luksoTestnet: getTestnetChainConfig(),
  },
};

export default config;
```

</file>

<file>

path: /package.json

url: https://github.com/CJ42/LUKSO-Hardhat-template/blob/main/package.json

```
{
  "name": "LUKSO-Hardhat-Template",
  "description": "Example repo for Hackathon to build a NFT collection using LSP7 or LSP8",
  "scripts": {
    "clean": "hardhat clean",
    "build": "hardhat compile"
  },
  "devDependencies": {
    "@nomicfoundation/hardhat-chai-matchers": "^2.0.0",
    "@nomicfoundation/hardhat-ethers": "^3.0.0",
    "@nomicfoundation/hardhat-network-helpers": "^1.0.0",
    "@nomicfoundation/hardhat-toolbox": "^3.0.0",
    "@nomicfoundation/hardhat-verify": "^1.0.0",
    "@typechain/ethers-v6": "^0.4.0",
    "@typechain/hardhat": "^8.0.0",
    "@types/chai": "^4.2.0",
    "@types/mocha": ">=9.1.0",
    "@types/node": ">=16.0.0",
    "chai": "^4.2.0",
    "dotenv": "^16.3.1",
    "ethers": "^6.4.0",
    "hardhat": "^2.18.2",
    "hardhat-gas-reporter": "^1.0.8",
    "prettier-plugin-solidity": "^1.1.3",
    "solidity-coverage": "^0.8.0",
    "ts-node": ">=8.0.0",
    "typechain": "^8.1.0",
    "typescript": ">=4.5.0"
  },
  "dependencies": {
    "@erc725/smart-contracts": "^6.0.0",
    "@lukso/lsp-factory.js": "^3.1.1",
    "@lukso/lsp-smart-contracts": "^0.12.0-rc.0"
  }
}
```

</file>

<file>

path: /scripts/DeployAndSetLSP8MetadataBaseURI.ts

url: https://github.com/CJ42/LUKSO-Hardhat-template/blob/main/scripts/DeployAndSetLSP8MetadataBaseURI.ts

```
import { ethers } from "hardhat";

import {BasicNFTCollection, BasicNFTCollection__factory} from "../typechain-types";

const LSP8TokenMetadataBaseURI =
  '0x1a7628600c3bac7101f53697f48df381ddc36b9015e7d7c9c5633d1252aa2843';

async function deployAndSetLSP8MetadataBaseURI() {
    const accounts = await ethers.getSigners();
    const deployer = accounts[0];

    const boredApesCollection: BasicNFTCollection = await new BasicNFTCollection__factory(
        deployer,
      ).deploy('BoredApeYachtClub', 'BAYC', deployer.address);

      await boredApesCollection.setData(
        LSP8TokenMetadataBaseURI,
        ethers.concat(
            //  0x6f357c6a  represents the hash function identifier,
            // the first 4 bytes of keccak256('keccak256(utf8)')
            // to be used to ensure that the metadata of the NFT is set in stone and cannot be changed
            // (for verifiability purpose)
            ["0x6f357c6a", ethers.toUtf8Bytes('ipfs://your-base-uri-on-ipf-goes-here/')]
        )
      );
}

deployAndSetLSP8MetadataBaseURI().catch((error) => {
    console.error(error);
    process.exitCode = 1;
  });
```

</file>

<file>

path: /scripts/deploy.ts

url: https://github.com/CJ42/LUKSO-Hardhat-template/blob/main/scripts/deploy.ts

```
import { ethers } from "hardhat";

import {BasicNFTCollection, BasicNFTCollection__factory} from "../typechain-types";

async function deployLSP8Collection() {
  const accounts = await ethers.getSigners();
  const deployer = accounts[0];

  const nftCollection: BasicNFTCollection = await new BasicNFTCollection__factory(deployer).deploy(
    "NFT Collection Name", // collection name
    "NFT", // collection symbol
    deployer.address
  );

}

deployLSP8Collection().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

</file>

<file>

path: /scripts/mintTickets.ts

url: https://github.com/CJ42/LUKSO-Hardhat-template/blob/main/scripts/mintTickets.ts

```
import { ethers } from "hardhat";

import {
    EventTicketsNFT,
    EventTicketsNFT__factory
} from "../typechain-types";

async function deployAndCreateTickets() {
    const accounts = await ethers.getSigners();
    const deployer = accounts[0];

    const luksoMeetupTickets: EventTicketsNFT = await new EventTicketsNFT__factory(
        deployer
    ).deploy(
        "LUKSO Meetup #2",
        "MUP2",
        deployer.address,
    )

    // create 100 entry tickets.
    // Give them to the deployer initially, who will distribute them afterwards.
    await luksoMeetupTickets.mint(
        deployer.address, // recipient
        100, // amount
        true, // force sending to an EOA
        "minting 100 tickets" // data
    );
}

deployAndCreateTickets().catch((error) => {
    console.error(error);
    process.exitCode = 1;
 });
```

</file>

<file>

path: /scripts/setDigitalStickerMetadata.ts

url: https://github.com/CJ42/LUKSO-Hardhat-template/blob/main/scripts/setDigitalStickerMetadata.ts

```
import { LSPFactory } from '@lukso/lsp-factory.js';
import { ethers } from 'hardhat';

import * as dotenv from "dotenv"
dotenv.config()

const provider = 'https://rpc.testnet.lukso.network';

const lspFactory = new LSPFactory(provider, {
  deployKey: process.env.PRIVATE_KEY, // Private key of the account which will deploy smart contracts
  chainId: 4201,
});

async function setStickerMetadata() {
    const accounts = await ethers.getSigners()
    const signer = accounts[0]

    console.log(process.env.PRIVATE_KEY)
    console.log(signer)
}

setStickerMetadata().catch((error) => {
    console.error(error);
    process.exitCode = 1;
  });
```

</file>

<file>

path: /test/NFTCollection.ts

url: https://github.com/CJ42/LUKSO-Hardhat-template/blob/main/test/NFTCollection.ts

```
import { time, loadFixture } from '@nomicfoundation/hardhat-toolbox/network-helpers';
import { anyValue } from '@nomicfoundation/hardhat-chai-matchers/withArgs';
import { expect } from 'chai';
import { ethers } from 'hardhat';

import { BasicNFTCollection, BasicNFTCollection__factory } from '../typechain-types';

const LSP8MetadataTokenURI = '0x1339e76a390b7b9ec9010000';
const LSP8ReferenceContract =
'0x708e7b881795f2e6b6c2752108c177ec89248458de3bf69d0d43480b3e5034e6';
const LSP8TokenMetadataBaseURI =
  '0x1a7628600c3bac7101f53697f48df381ddc36b9015e7d7c9c5633d1252aa2843';

describe('BoredApes Collection', () => {
  it('test', async () => {
    const accounts = await ethers.getSigners();
    const deployer = accounts[0];

    const boredApesCollection: BasicNFTCollection = await new BasicNFTCollection__factory(
      deployer,
    ).deploy('BoredApeYachtClub', 'BAYC', deployer.address);

    await boredApesCollection.setData(
      LSP8TokenMetadataBaseURI,
      ethers.toUtf8Bytes('ipfs://your-base-uri-on-ipf-goes-here/'),
    );

    expect(await boredApesCollection.getData(LSP8TokenMetadataBaseURI)).to.equal(
      ethers.toUtf8Bytes('ipfs://your-base-uri-on-ipf-goes-here/'),
    );
  });
});
```

</file>

<file>

path: /tsconfig.json

url: https://github.com/CJ42/LUKSO-Hardhat-template/blob/main/tsconfig.json

```
{
  "compilerOptions": {
    "target": "es2020",
    "module": "commonjs",
    "esModuleInterop": true,
    "forceConsistentCasingInFileNames": true,
    "strict": true,
    "skipLibCheck": true,
    "resolveJsonModule": true
  }
}
```

</file>

</repository>

<repository>

twitter-up-dapp

overview

repository: https://github.com/CJ42/twitter-up-dapp/
userName: CJ42
repository: twitter-up-dapp
branch: main
date: 2023-11-27T10:50:57+01:00 (1701078657)

<file>

path: /README.md

url: https://github.com/CJ42/twitter-up-dapp/blob/main/README.md

LUKSO Twitter UP dApp

Based on [nextjs-twitter-starter (https://github.com/Dineshs91/nextjs-twitter-starter)]

Setup

1. Install the dependencies

```
yarn install
```

2. in the Twitter Developer dashboard, create a project with Twitter API v2 and save the credentials of the projects under the .env.local file as shown below:

```
TWITTER_AUTH_CLIENT_ID="..."
TWITTER_AUTH_CLIENT_SECRET="..."
```

3. add the callback url of the project as http://localhost:3000/twitter-callback in the Twitter Developer dashboard.

Running the dApp

```
yarn dev
```

Navigate to [localhost:3000 (https://localhost:3000)] and follow the steps.

Potential future improvements

- [] In the UP Profile metadata, add the following extra infos
- location

- public_metrics
- created_at
- entities?
- [x] Find a way to download banner image (not available in twitter API free tier)
  https://twittercommunity.com/t/how-to-get-banner-url-for-a-user/168692/7

Already searched and banner image only available with Basic paid plan.
</file>

<file>

path: /next-env.d.ts

url: https://github.com/CJ42/twitter-up-dapp/blob/main/next-env.d.ts

```
/// <reference types="next" />
/// <reference types="next/image-types/global" />

// NOTE: This file should not be edited
// see https://nextjs.org/docs/basic-features/typescript for more information.
```

</file>

<file>

path: /next.config.js

url: https://github.com/CJ42/twitter-up-dapp/blob/main/next.config.js

```
module.exports = {
  reactStrictMode: true,
  async rewrites() {
    return [
      // Rewrite everything else to use  pages/index 
      {
        source: '/:path*',
        destination: '/',
      },
    ];
  },
}
```

</file>

<file>

path: /package.json

url: https://github.com/CJ42/twitter-up-dapp/blob/main/package.json

```
{
  "name": "nextjs-twitter-starter",
  "version": "0.1.0",
  "private": false,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint"
  },
  "dependencies": {
    "@emotion/react": "^11.11.0",
    "@emotion/server": "^11.11.0",
    "@erc725/erc725.js": "^0.17.2",
    "@lukso/lsp-factory.js": "^2.5.1",
    "@lukso/lsp-smart-contracts": "^0.10.0",
    "@lukso/web-components": "^1.30.0",
    "@mantine/core": "^6.0.10",
    "@mantine/hooks": "^6.0.10",
    "@mantine/next": "^6.0.10",
    "@mantine/prism": "^6.0.10",
    "autoprefixer": "^10.3.7",
    "bulma": "^0.9.4",
    "ethers": "~5.7.0",
    "next": "^13.4.0",
    "node-sass": "~7.0.0",
    "postcss": "^8.4.4",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-router-dom": "^6.11.1",
    "sass": "^1.66.1",
    "tailwindcss": "^3.0.1",
    "twitter-api-sdk": "^1.2.1"
  },
  "devDependencies": {
    "@types/node": "^20.0.0",
    "@types/react": "^17.0.0",
    "typescript": "^5.0.4"
  }
}
```

</file>

<file>

path: /pages/LSP3Profile-template.json

url: https://github.com/CJ42/twitter-up-dapp/blob/main/pages/LSP3Profile-template.json

```
{
  "LSP3Profile": {
    "name": "",
    "description": "",
    "links": [
      { "title": "Twitter", "url": "" }
    ],
    "tags": ["public profile", "Twitter"],
    "profileImage": []
  }
}
```

</file>

<file>

path: /pages/LSP3Profile.json

url: https://github.com/CJ42/twitter-up-dapp/blob/main/pages/LSP3Profile.json

```
{
  "LSP3Profile": {
    "name": "frozeman",
    "description": "The inventor of ERC725 and ERC20...",
    "links": [
      { "title": "Twitter", "url": "https://twitter.com/feindura" },
      { "title": "lukso.network", "url": "https://lukso.network" }
    ],
    "tags": ["brand", "public profile"],
    "avatar": [
      {
        "hashFunction": "keccak256(bytes)",
        "hash": "0x98fe032f81c43426fbcfb21c780c879667a08e2a65e8ae38027d4d61cdfe6f55",
        "url": "ifps://QmPJESHbVkPtSaHntNVY5F6JDLW8v69M2d6khXEYGUMn7N",
        "fileType": "fbx"
      }
    ],
    "profileImage": [
      {
        "width": 1800,
        "height": 1013,
        "hashFunction": "keccak256(bytes)",
        "hash": "0x98fe032f81c43426fbcfb21c780c879667a08e2a65e8ae38027d4d61cdfe6f55",
        "url": "ifps://QmPJESHbVkPtSaHntNVY5F6JDLW8v69M2d6khXEYGUMn7N"
      },
      {
        "address": "0x1231c7436a77a009a97e48e4e10c92e89fd95fe15",
        "tokenId": "0xdde1c7436a77a009a97e48e4e10c92e89fd95fe1556fc5c62ecef57cea51aa37"
      }
    ],
    "backgroundImage": [
      {
        "width": 1800,
        "height": 1013,
        "hashFunction": "keccak256(bytes)",
        "hash": "0x98fe032f81c43426fbcfb21c780c879667a08e2a65e8ae38027d4d61cdfe6f55",
        "url": "ifps://QmPJESHbVkPtSaHntNVY5F6JDLW8v69M2d6khXEYGUMn7N"
      },
      {
        "width": 1024,
        "height": 576,
        "hashFunction": "keccak256(bytes)",
        "hash": "0xfce1c7436a77a009a97e48e4e10c92e89fd95fe1556fc5c62ecef57cea51aa37",
        "url": "ifps://QmZc9uMJxyUeUpuowJ7AD6MKoNTaWdVNcBj72iisRyM9Su"
      }
    ]
  }
}
```


</file>

<file>

path: /pages/_app.tsx

url: https://github.com/CJ42/twitter-up-dapp/blob/main/pages/_app.tsx

```
import '../styles/globals.css'

import { AppProps } from 'next/app';

function MyApp({ Component, pageProps }: AppProps) {
  return <Component {...pageProps}/>
}

export default MyApp
```

</file>

<file>

path: /pages/api/twitter-banner.ts

url: https://github.com/CJ42/twitter-up-dapp/blob/main/pages/api/twitter-banner.ts

```
import Client from "twitter-api-sdk";
import {
  createTwitterOAuthUser,
  getTwitterAuthURL,
} from "../../services/twitter";

export default async (req: any, res: any) => {
  const authClient = createTwitterOAuthUser();
  getTwitterAuthURL(authClient);

  const token = await authClient.requestAccessToken(req.body.authCode);

    // TODO: check that we have received a token
    const twitterApi = new Client(authClient);

    const result = await twitterApi.users;

  try {
  } catch (error) {
    console.error(error);
    res.status(500).json("error occurred when fetching twitter banner image");
  }
};
```

</file>

<file>

path: /pages/api/twitter-user.ts

url: https://github.com/CJ42/twitter-up-dapp/blob/main/pages/api/twitter-user.ts

```
import Client from "twitter-api-sdk";
import {
  createTwitterOAuthUser,
  getTwitterAuthURL,
} from "../../services/twitter";

export default async (req: any, res: any) => {
  const authClient = createTwitterOAuthUser();
  getTwitterAuthURL(authClient);

  if (!req.body) {
    res.status(400).json({ message: "invalid request body" });
  }

  // 5. use the &amp;nbsp;code&amp;nbsp; to get the access token
  try {
    const token = await authClient.requestAccessToken(req.body.authCode);

    // TODO: check that we have received a token
    const twitterApi = new Client(authClient);

    const userInfos = await twitterApi.users.findMyUser({
      // A comma separated list of User fields to display
      "user.fields": [
        "created_at",
        "description",
        "entities",
        "id",
        "location",
        "name",
        "pinned_tweet_id",
        "profile_image_url",
        "protected",
        "public_metrics",
        "url",
        "username",
        "verified",
        "withheld",
      ],

      // A comma separated list of Tweet fields to display.
      "tweet.fields": [
        "attachments",
        "author_id",
        "context_annotations",
```

```
      "conversation_id",
      "created_at",
      "edit_controls",
      "entities",
      "geo",
      "id",
      "in_reply_to_user_id",
      "lang",
      "non_public_metrics",
      "public_metrics",
      "organic_metrics",
      "promoted_metrics",
      "possibly_sensitive",
      "referenced_tweets",
      "reply_settings",
      "source",
      "text",
      "withheld",
    ],

    // A comma separated list of fields to expand
    expansions: ["pinned_tweet_id"],
  });

  console.log(userInfos);
  console.log(userInfos.data?.entities?.url?.urls);
  console.log(userInfos.data?.entities?.description);
  res.status(200).json(userInfos);
 } catch (error) {
  console.error(error);
  res.status(500).json("error occurred when fetching twitter user data");
 }
};
```

</file>

<file>

path: /pages/index.tsx

url: https://github.com/CJ42/twitter-up-dapp/blob/main/pages/index.tsx

```
import Head from 'next/head'

export default function Home() {
 return (
  <div className="mt-4 max-w-xs p-4 md:max-w-md lg:max-w-2xl xl:max-w-4xl mx-auto">
   <Head>
    <title>Import your Twitter Profile on your UP</title>
    <meta name="description" content="Next Twitter Starter" />
    <link rel="icon" href="/favicon.ico" />
```

```
    </Head>

    <main className="mt-10">
      <h1 className="text-3xl font-bold">
        Import your Twitter Profile on your UP.
      </h1>

      <a href="/twitter" className="flex items-center p-8 mt-8 border rounded-md space-x-4 hover:bg-gray-
100">
        <p className="font-semibold text-lg">Let's get Started &rarr;</p>
      </a>
{/*
      <div className="p-8 mt-8 border rounded-md space-y-4">
      <div>
          <p className="font-semibold text-lg">LSP3 Profile Metadata</p>
          <p>
            This is what the JSON under your
            <code className="">LSP3Profile</code>
            metadata looks like now
          </p>
        </div>
        <pre>
          "some text"
        </pre>
      </div> */}

      {/* <div className="p-8 mt-8 border rounded-md space-y-4">
        <div>
          <p className="font-semibold text-lg">Pages</p>
          <p>
            Get started by editing{' '}
            <code className="">pages/index.js</code>
          </p>
        </div>
        <div>
          <p className="font-semibold text-lg">API</p>
          <p>
            Get started with Twitter API by looking at{' '}
            <code className="">pages/api/twitter-user.js</code>
          </p>
        </div>
      </div> */}

      {/* <div className="flex mt-8 flex-wrap gap-4">
        <a href="https://nextjs.org/docs" className="flex-auto border rounded-md px-4 py-6 hover:bg-gray-
100">
          <h2 className="font-semibold text-lg">Next.js Documentation &rarr;</h2>
          <p>Find in-depth information about Next.js features and API.</p>
        </a>

        <a href="https://github.com/draftbit/twitter-lite" className="flex-auto border rounded-md px-4 py-6
hover:bg-gray-100">
```

```
          <h2 className="font-semibold text-lg">Twitter-lite Documentation &rarr;</h2>
          <p>Find Twitter-lite documentation.</p>
        </a>

        <a href="https://tailwindcss.com/docs" className="flex-auto border rounded-md px-4 py-6 hover:bg-gray-100">
          <h2 className="font-semibold text-lg">Tailwindcss Documentation &rarr;</h2>
          <p>Find in-depth information about Tailwindcss features.</p>
        </a>

        <a href="https://developer.twitter.com/en/docs/twitter-api/v1" className="flex-auto border rounded-md px-4 py-6 hover:bg-gray-100">
          <h2 className="font-semibold text-lg">Twitter API Documentation &rarr;</h2>
          <p>Checkout Twitter API documentation.</p>
        </a>

        <a
          href="https://vercel.com/new?utm_source=create-next-app&utm_medium=default-template&utm_campaign=create-next-app"
          className="flex-auto border rounded-md px-4 py-6 hover:bg-gray-100"
        >
          <h2 className="font-semibold text-lg">Deploy &rarr;</h2>
          <p>
            Instantly deploy your Next.js site to a public URL with Vercel.
          </p>
        </a>
      </div> */}
    </main>
  </div>
 )
}
```

</file>

<file>

path: /pages/twitter-callback.tsx

url: https://github.com/CJ42/twitter-up-dapp/blob/main/pages/twitter-callback.tsx

```
import { useState, useEffect } from "react";
import { useRouter } from "next/router";
import { Prism } from "@mantine/prism";
import { ethers } from "ethers";

import UniversalProfile from "@lukso/lsp-smart-contracts/artifacts/UniversalProfile.json";
import ERC725JS, { ERC725JSONSchema } from "@erc725/erc725.js";
import { LSPFactory, ProfileDataBeforeUpload } from "@lukso/lsp-factory.js";

declare global {
  interface Window {
```

```
    ethereum?: any;
  }
}

// Twitter brings us back to this page after OAuth2 authorization
export default function TwitterCallback(props: any) {
  // Get the &amp;nbsp;code&amp;nbsp; from the url query params
  const router = useRouter();
  const { code } = router.query;

  const [provider, setProvider] = useState<ethers.providers.Web3Provider>({} as
ethers.providers.Web3Provider);
  const [universalProfileAddress, setUniversalProfileAddress] = useState("");
  const [lsp3ProfileDataIPFSURL, setLsp3ProfileDataIPFSURL] = useState("");
  const [lsp3ProfileDataJSON, setLSP3ProfileDataJSON] = useState<ProfileDataBeforeUpload>(
    {} as ProfileDataBeforeUpload
  );

  useEffect(() => {
    if (window.ethereum) {
      const etherProvider = new ethers.providers.Web3Provider(window.ethereum);
      setProvider(etherProvider);
    }
  }, []);

  const connectUniversalProfile = async () => {
    const accountsRequest: string[] = await provider.send("eth_requestAccounts", []);
    setUniversalProfileAddress(accountsRequest[0]);

    // debug
    console.log("Universal Profile's address: ", accountsRequest[0]);
  };

  const getTwitterUserInfos = async () => {
    // Call the backend API endpoint with the &amp;nbsp;code&amp;nbsp; to query user's Twitter profile data
    fetch("/api/twitter-user", {
      method: "POST",
      body: JSON.stringify({ authCode: code }),
      headers: {
        "Content-Type": "application/json",
      },
    })
      .then((res) => res.json())
      .then(async (data) => {
        const { data: profileData } = data;
        const preparedData = await prepareProfileDetails(profileData);
        setLSP3ProfileDataJSON(preparedData);

        // debug
        console.log("      Data received from Twitter API: ", data);
        console.log("      Prepared the LSP3 Profile Metadata JSON: ", preparedData);
      });
```

```
};

// Prepare the LSP3Profile JSON template by filling up the fields
const prepareProfileDetails = async (twitterUserData: any): Promise<ProfileDataBeforeUpload> => {
  const twitterUsername = twitterUserData.username;

  // Upload images to ipfs using lsp-factory.js and add them in the list below
  const lspFactory = new LSPFactory(provider);

  try {
    // 1. download the profile image from twitter
    const response = await fetch(twitterUserData.profile_image_url);
    const imageBlob = await response.blob();

    const profileImage: File = new File([imageBlob], "profile", {
      type:  image/${imageBlob.type} ,
    });

    // Upload our JSON file to IPFS
    const uploadResult = await lspFactory.UniversalProfile.uploadProfileData({
      name: twitterUserData.name,
      description: twitterUserData.description,
      profileImage: profileImage,
      tags: ["Twitter", "UniversalProfile", twitterUsername], // Add Twitter user handle as a tag
      links: [{ title: "Twitter", url: "https://twitter.com/" + twitterUsername }],
    });

    // e.g: ipfs://QmQytqrSB5JXygk2PGqFVpeVjduzS4ewQd3F9hd53XCmGZ
    // can be viewed at:
https://2eff.lukso.dev/ipfs/QmQytqrSB5JXygk2PGqFVpeVjduzS4ewQd3F9hd53XCmGZ
    const lsp3ProfileIPFSUrl = uploadResult.url;

    const ipfsResponse = await fetch(
      "https://2eff.lukso.dev/ipfs/" + lsp3ProfileIPFSUrl.substring(7) // strip the ipfs:// prefix
    );
    const jsonResponse = await ipfsResponse.json();

    console.log("      Successfully uploaded on IPFS: ", lsp3ProfileIPFSUrl);
    console.log("      LSP3Profile JSON including IPFS links: ", jsonResponse);

    setLsp3ProfileDataIPFSURL(lsp3ProfileIPFSUrl);

    return jsonResponse;
  } catch (error) {
    throw new Error( Error while uploading profile metadata to IPFS: ${error} );
  }
};

/**
 * @dev Create an instance of the UniversalProfile smart contract
 * using the UniversalProfile ABI and bytecode from the  @lukso/lsp-smart-contracts  package.
 */
```

```
 async function createUniversalProfileInstance() {
   return new ethers.ContractFactory(UniversalProfile.abi, UniversalProfile.bytecode,
provider.getSigner()).attach(
     universalProfileAddress
   );
 }

 // Schema of LSP3Profile Metadata key
 const LSP3ProfileSchema: ERC725JSONSchema = {
   name: "LSP3Profile",
   key: "0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5",
   keyType: "Singleton",
   valueType: "bytes",
   valueContent: "JSONURL",
 };

 /**
  * @dev Update the LSP3Profile metadata of the Universal Profile
  * by calling the  setData  function of the UniversalProfile smart contract
  * on the LSP3Profile data key.
  *
  * This function will also prepare the data to be set by encoding it with erc725.js
  */
 async function updateLSP3ProfileMetadata() {
   // Create an instance of ERC725.js, with the schema of LSP3Profile
   const erc725js = new ERC725JS([LSP3ProfileSchema], universalProfileAddress, provider, {
     ipfsGateway: "https://2eff.lukso.dev/ipfs/",
   });

   const encodedData = await erc725js.encodeData([
     {
       keyName: LSP3ProfileSchema.name,
       value: {
         hashFunction: "keccak256(utf8)",
         hash: ethers.utils.keccak256(ethers.utils.toUtf8Bytes(JSON.stringify(lsp3ProfileDataJSON))),
         url: lsp3ProfileDataIPFSURL,
       },
     },
   ]);

   const profileInstance = await createUniversalProfileInstance();

   console.log("      Successfully encoded with erc725.js: ", encodedData);

   // We are only updating one data key here
   const tx = await profileInstance.setData(encodedData.keys[0], encodedData.values[0], {
     gasLimit: 10_000_000,
     from: universalProfileAddress,
   });

   console.log("tx:", tx);
 }
```

```
const clearProfileDetails = async () => {
  const profileInstance = await createUniversalProfileInstance();

  await profileInstance.setData(LSP3ProfileSchema.key, "0x", {
    gasLimit: 1_000_000,
    from: universalProfileAddress,
  });
};

return (
  <div className="p-4">
    <h2 className="text-2xl font-semibold">Successfully logged-in to Twitter!</h2>

    <p className="mt-8 mb-8">Let's connect your UP now      </p>

    <div className="grid grid-cols-2">
      <div className="first-column">
        <h2 className="mb-4 text-2xl">Step 1 - Connect your       to this dApp</h2>

        <button
          className="bg-purple-500 hover:bg-purple-700 text-white font-bold py-2 px-4 rounded-full"
          onClick={() => connectUniversalProfile()}
        >
              Connect my UP
        </button>

        <p className="mt-2">
          {universalProfileAddress ? (
            <span className="text-sm text-gray-500">
                  Connected to UP at: <code>{universalProfileAddress}</code>
            </span>
          ) : (
            <span className="text-sm text-gray-500">      Not connected to UP yet</span>
          )}
        </p>

        <h2 className="mt-8 mb-4 text-2xl">Step 2 - Fetch your Twitter Profile infos</h2>
        <button
          className="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded-full"
          onClick={() => getTwitterUserInfos()}
        >
              Fetch Twitter User Infos
        </button>
        <p className="mt-2">
          Once your Twitter profile info has been fetched, you will see it appearing in the
<code>LSP3Profile</code> Metadata below ⬇
        </p>

        <h2 className="mt-8 mb-4 text-2xl">Step 3 - Import your Twitter infos to your profile</h2>
        <button
          className="bg-green-500 hover:bg-green-700 text-white font-bold py-2 px-4 rounded-full"
```

```
          onClick={() => updateLSP3ProfileMetadata()}
        >
              Upload Twitter info to my UP
        </button>
      </div>

      <div className="second-column">
        <button
          className="bg-red-500 hover:bg-red-700 text-white font-bold py-2 px-4 rounded-full"
          onClick={() => clearProfileDetails()}
        >
          Clear UP profile details
        </button>
        {/* <button>
          <a href="/">Go back home</a>
        </button> */}
      </div>
    </div>

    <hr className="mt-8 mb-8" />

    <h2 className="text-1sm italic">Generated LSP3Profile JSON Metadata</h2>
    <div className="mt-8">
      <Prism language="json" withLineNumbers scrollAreaComponent="div">
        {JSON.stringify(lsp3ProfileDataJSON, null, 4)}
      </Prism>
    </div>
  </div>
 );
}
```

</file>

<file>

path: /pages/twitter.tsx

url: https://github.com/CJ42/twitter-up-dapp/blob/main/pages/twitter.tsx

```
import { createTwitterOAuthUser, getTwitterAuthURL } from '../services/twitter'
import { useRouter } from 'next/router'

export default function Twitter(props: any) {
  const router = useRouter();

  return (
     <div className="p-2 max-w-xs lg:max-w-md mx-auto mt-12">
       <h2 className="text-2xl font-semibold">Let's connect your Twitter account</h2>

       <div className="mt-8">
         <p>Click the button below to connect to your Twitter account.</p>
         <p className="mt-2">We'll collect information from your Twitter profile and import them in your UP.
</p>
       </div>

       <div className="mt-8">
         <div className="mt-8">
           <button className="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded-full"
onClick={async () => {
             // 1. redirect to authURL
             const authClient = createTwitterOAuthUser();
             const authUrl = await getTwitterAuthURL(authClient);
             router.push(authUrl)
           }}>
              Connect Twitter Account
           </button>
         </div>
       </div>
     </div>
  )
}

export async function getServerSideProps(context: any) {
  let twitterHandle = process.env.TEST_TWITTER_HANDLE
  let twitterInfo = null

  return {
    props: {
      "twitterInfo": null
    },
  }
}
```

</file>

<file>

path: /postcss.config.js

url: https://github.com/CJ42/twitter-up-dapp/blob/main/postcss.config.js

```
module.exports = {
  plugins: {
    tailwindcss: {},
    autoprefixer: {},
  },
}
```

</file>

<file>

path: /public_metrics.json

url: https://github.com/CJ42/twitter-up-dapp/blob/main/public_metrics.json


```
{
  "data": {
    "name": "Jean Cavallera - CJ42",
    "public_metrics": {
      "followers_count": 1358,
      "following_count": 559,
      "tweet_count": 637,
      "listed_count": 25
    },
    "id": "3382868896",
    "username": "JeanCavallera"
  }
}
```


</file>

<file>

path: /services/twitter.ts

url: https://github.com/CJ42/twitter-up-dapp/blob/main/services/twitter.ts

```
import { auth } from 'twitter-api-sdk';
import { OAuth2User, OAuth2Scopes } from 'twitter-api-sdk/dist/OAuth2User';

const twitterAuthScopes: OAuth2Scopes[] = ['users.read', 'tweet.read']

export const createTwitterOAuthUser = () => {
  return new auth.OAuth2User({
    client_id: "ckxFaHlnX2FUdDdZMENVazBFMUk6MTpjaQ",
    client_secret: process.env.TWITTER_AUTH_CLIENT_SECRET,
    callback: "http://twitter-up-dapp.vercel.app/twitter-callback",
    scopes: twitterAuthScopes,
  })
}

export const getTwitterAuthURL = async (authClient: OAuth2User) => {
  const authUrl = authClient.generateAuthURL({
    state: "4d97327ce8d64acfa53f",
    code_challenge_method: 'plain',
    code_challenge: "1a5489693c94ff27e998",
  });

  return authUrl;
}
```

</file>

<file>

path: /styles/globals.css

url: https://github.com/CJ42/twitter-up-dapp/blob/main/styles/globals.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

</file>

<file>

path: /tailwind.config.js

url: https://github.com/CJ42/twitter-up-dapp/blob/main/tailwind.config.js

```
module.exports = {
  purge: ['./pages/**/*.{js,ts,jsx,tsx}', './components/**/*.{js,ts,jsx,tsx}'],
  presets: [require('@lukso/web-components/tailwind.config')],
  darkMode: false, // or 'media' or 'class'
  theme: {
    extend: {},
  },
  variants: {
    extend: {},
  },
  plugins: [],
}
```

</file>

<file>

path: /tsconfig.json

url: https://github.com/CJ42/twitter-up-dapp/blob/main/tsconfig.json

```
{
  "compilerOptions": {
    "lib": [
      "dom",
      "dom.iterable",
      "esnext"
    ],
    "allowJs": true,
    "skipLibCheck": true,
    "strict": true,
    "forceConsistentCasingInFileNames": true,
    "noEmit": true,
    "incremental": true,
    "esModuleInterop": true,
    "module": "esnext",
    "moduleResolution": "node",
    "resolveJsonModule": true,
    "isolatedModules": true,
    "jsx": "preserve"
  },
  "include": [
    "next-env.d.ts",
    "**/*.ts",
    "**/*.tsx"
  ],
  "exclude": [
    "node_modules"
  ]
}
```

</file>

<file>

path: /user_data.json

url: https://github.com/CJ42/twitter-up-dapp/blob/main/user_data.json

```
data: {
   protected: false,
   location: 'London, England',
   username: 'JeanCavallera',
   description: 'Smart Contract Engineer at @lukso_io       | Author of “All About Solidity” articles series
    ',
   public_metrics: {
     followers_count: 1358,
     following_count: 560,
     tweet_count: 637,
     listed_count: 25
   },
   profile_image_url: 'https://pbs.twimg.com/profile_images/1628818297041960961/y09nez7F_normal.jpg',
   verified: false,
   created_at: '2015-07-19T08:40:56.000Z',
   name: 'Jean Cavallera - CJ42',
   id: '3382868896',
   entities: { url: [Object], description: [Object] },
   url: 'https://t.co/4W1qQCrXNm'
 }
```

</file>

<file>

path: /user_mentions.json

url: https://github.com/CJ42/twitter-up-dapp/blob/main/user_mentions.json

```
{ mentions: [ { start: 27, end: 36, username: 'lukso_io' } ] }
```

</file>

<file>

path: /user_urls.json

url: https://github.com/CJ42/twitter-up-dapp/blob/main/user_urls.json

```
[
  {
    start: 0,
    end: 23,
    url: 'https://t.co/4W1qQCrXNm',
    expanded_url: 'https://link.medium.com/hWecsU127xb',
    display_url: 'link.medium.com/hWecsU127xb'
  }
]
```

</file>

</repository>

<repository>

erc725.js

overview

repository: https://github.com/ERC725Alliance/erc725.js/
userName: ERC725Alliance
repository: erc725.js
branch: main
date: 2023-11-27T10:50:57+01:00 (1701078657)

<file>

path: /.eslintrc

url: https://github.com/ERC725Alliance/erc725.js/blob/main/.eslintrc

```
{
  "env": {
    "browser": true,
    "es2021": true,
    "node": true,
    "mocha": true
  },
  "parserOptions": {
    "ecmaVersion": 12,
    "sourceType": "module"
  },
  "parser": "@typescript-eslint/parser",
  "plugins": ["prettier", "@typescript-eslint"],
  "extends": [
    "airbnb-base",
    "eslint:recommended",
    "plugin:@typescript-eslint/recommended",
    "prettier",
    "plugin:prettier/recommended"
  ],
  "settings": {
    "import/resolver": {
      "node": {
        "extensions": [".js", ".jsx", ".ts", ".tsx"],
        "moduleDirectory": ["node_modules"],
        "paths": ["."]
      }
    }
  },
  "rules": {
    "prettier/prettier": "error",
    // TYPESCRIPT
```

```
    "@typescript-eslint/no-shadow": ["error"],
    "@typescript-eslint/explicit-function-return-type": "off",
    "@typescript-eslint/explicit-module-boundary-types": "off",
    "@typescript-eslint/no-use-before-define": ["error"],
    "@typescript-eslint/no-explicit-any": "off",
    "@typescript-eslint/ban-ts-ignore": "off",
    "@typescript-eslint/ban-ts-comment": "off",
    // OTHER
    "class-methods-use-this": "off",
    "no-shadow": "off",
    "no-console": "off",
    "import/extensions": ["error", "always", { "ts": "never", "js": "never" }],
    "max-classes-per-file": ["error", 3],
    "no-underscore-dangle": ["error", { "allowAfterThis": true }],
    "import/no-extraneous-dependencies": [
      "error",
      {
        "devDependencies": true,
        "optionalDependencies": false,
        "peerDependencies": false
      }
    ],
    "no-plusplus": ["error", { "allowForLoopAfterthoughts": true }],
    "lines-between-class-members": "off",
    "prefer-template": 0, // Allow simple string concatenation
    "no-await-in-loop": 0, // NOTE: This should be removed?
    "import/prefer-default-export": 0,
    "arrow-body-style": "off",
    "no-bitwise": "off"
  },
  "ignorePatterns": ["node_modules/**/*", "build/**/*"]
}
```

</file>

<file>

path: /.github/ISSUE_TEMPLATE.md

url: https://github.com/ERC725Alliance/erc725.js/blob/main/.github/ISSUE_TEMPLATE.md

I'm submitting a...

- [ ] bug report
- [ ] feature request
- [ ] question about the decisions made in the repository
- [ ] question about how to use this project

Summary

Other information (e.g. detailed explanation, stack traces, related issues, suggestions how to fix, links for us to have context, eg. StackOverflow, personal fork, etc.)

</file>

<file>

path: /.github/PULL_REQUEST_TEMPLATE.md

url: https://github.com/ERC725Alliance/erc725.js/blob/main/.github/PULL_REQUEST_TEMPLATE.md

What kind of change does this PR introduce (bug fix, feature, docs update, ...)?

What is the current behaviour (you can also link to an open issue here)?

What is the new behaviour (if this is a feature change)?

Other information:

</file>

<file>

path: /.mocharc.json

url: https://github.com/ERC725Alliance/erc725.js/blob/main/.mocharc.json

```
{
  "extension": ["ts"],
  "spec": "src/**/*.test.ts",
  "require": "ts-node/register",
  "timeout": 20000
}
```

</file>

<file>

path: /.node-version

url: https://github.com/ERC725Alliance/erc725.js/blob/main/.node-version

```
14.17.0
```

</file>

<file>

path: /.npmignore

url: https://github.com/ERC725Alliance/erc725.js/blob/main/.npmignore

```
tsconfig.json
tsconfig.module.json
tsconfig.tsbuildinfo
tsconfig.module.tsbuildinfo
requirements.txt
src
test
docs
.idea
.vscode
.github
examples
.nyc_output/
generatedSchema.ts
.mocharc.json
.prettierrc
.node-version
.eslintrc
coverage
```

</file>

<file>

path: /.prettierrc

url: https://github.com/ERC725Alliance/erc725.js/blob/main/.prettierrc

```
{
  "trailingComma": "all",
  "tabWidth": 2,
  "semi": true,
  "singleQuote": true
}
```

</file>

<file>

path: /CHANGELOG.md

url: https://github.com/ERC725Alliance/erc725.js/blob/main/CHANGELOG.md

Changelog

All notable changes to this project will be documented in this file. See [standard-version (https://github.com/conventional-changelog/standard-version)](https://github.com/conventional-changelog/standard-version) for commit guidelines.

0.21.2 (https://github.com/ERC725Alliance/erc725.js/compare/v0.21.1...v0.21.2) (2023-11-07)

Bug Fixes

- Add for unknown verification method to allow for null verification data in LSP2 (f205818 (https://github.com/ERC725Alliance/erc725.js/commit/f205818af348471bde8f88af2008497b8c13e25
- Add more fixes per PR comments (e7302e4 (https://github.com/ERC725Alliance/erc725.js/commit/e7302e4504408e2f4f6304badd2024bfe05fcf4
- Change to verification object (ddd2ab2 (https://github.com/ERC725Alliance/erc725.js/commit/ddd2ab23d1c5181745827f338d9abaea48c77
- More renames _FUNCTIONS to _METHODS (1a96be1 (https://github.com/ERC725Alliance/erc725.js/commit/1a96be1dd15942d2a844bc26b9ab73e053e3b
- Move @types/jest and jest (852918c (https://github.com/ERC725Alliance/erc725.js/commit/852918c72228b3839ba60730dadef66837008

0.21.1 (https://github.com/ERC725Alliance/erc725.js/compare/v0.21.0...v0.21.1) (2023-11-06)

Bug Fixes

- incorrect hex for LSP8MetadataTokenURI (0500a75 (https://github.com/ERC725Alliance/erc725.js/commit/0500a752e3117c5c7e9df8cfed22cb5d6fee20c

0.21.0 (https://github.com/ERC725Alliance/erc725.js/compare/v0.20.1...v0.21.0) (2023-11-02)

⚠ BREAKING CHANGES

- update lsp3/lsp4 verificationData

Features

- new gas parameter (82e3833 (https://github.com/ERC725Alliance/erc725.js/commit/82e383345a712619b5c6a1030b124d262511

Code Refactoring

- update lsp3/lsp4 verificationData (9640d9f (https://github.com/ERC725Alliance/erc725.js/commit/9640d9fbf88c7cf694b9e82cc3a711350334b0

0.20.1 (https://github.com/ERC725Alliance/erc725.js/compare/v0.20.0...v0.20.1) (2023-10-30)

Bug Fixes

- incorrect permission value for EXECUTE_RELAY_CALL (55b8f5e (https://github.com/ERC725Alliance/erc725.js/commit/55b8f5e64c29c5a85d872f605667c88c1546f6b

0.20.0 (https://github.com/ERC725Alliance/erc725.js/compare/v0.19.0...v0.20.0) (2023-10-18)

Features

- add permission EXECUTE_RELAY_CALL (6db8835 (https://github.com/ERC725Alliance/erc725.js/commit/6db8835ccd9d1082d9e8184bb2f14972760be

0.19.0 (https://github.com/ERC725Alliance/erc725.js/compare/v0.18.0...v0.19.0) (2023-10-05)

⚠ BREAKING CHANGES

- change LSP3 to SupportedStandards:LSP3Profile (#307 (https://github.com/ERC725Alliance/erc725.js/issues/307))
- new encoding for static value types (not arrays []) (#288 (https://github.com/ERC725Alliance/erc725.js/issues/288))
- change ArrayLength value from uint256 to uint128 (#287 (https://github.com/ERC725Alliance/erc725.js/issues/287))

Features

- add checkPermissions function (17d2258 (https://github.com/ERC725Alliance/erc725.js/commit/17d225843c236951ef1515a0ff91095b5ef27cc
- add schemas for LSP8 + LSP17 (#311 (https://github.com/ERC725Alliance/erc725.js/issues/311)) (1e8dbf7 (https://github.com/ERC725Alliance/erc725.js/commit/1e8dbf765c6c5e250539b402e9bd5a395966a

Bug Fixes

- decode any uint256 as number not string (#289 (https://github.com/ERC725Alliance/erc725.js/issues/289)) (37203f1 (https://github.com/ERC725Alliance/erc725.js/commit/37203f14d313a0caff75724dc74175c741c1b5
- dependencies & example (#302 (https://github.com/ERC725Alliance/erc725.js/issues/302)) (9979e89 (https://github.com/ERC725Alliance/erc725.js/commit/9979e89e438cd9f7cc586d7dc271de969f13b1
- incorrect value in schema for array length in ...Map (#310 (https://github.com/ERC725Alliance/erc725.js/issues/310)) (0d28b13 (https://github.com/ERC725Alliance/erc725.js/commit/0d28b1317dc085078090a8babacf4db517d91
- Remove hardcoded require (5279278 (https://github.com/ERC725Alliance/erc725.js/commit/527927812b1a05b13f8dc6b14aecaa6d24e98
- variable naming (44b4785

(https://github.com/ERC725Alliance/erc725.js/commit/44b47851ed63b817edc21c63655d67bac13a7

Code Refactoring

- change ArrayLength value from uint256 to uint128 (#287 (https://github.com/ERC725Alliance/erc725.js/issues/287)) (c95ee8a (https://github.com/ERC725Alliance/erc725.js/commit/c95ee8a53bf25bcf47777054af27cae1fbad8b2
- change LSP3 to SupportedStandards:LSP3Profile (#307 (https://github.com/ERC725Alliance/erc725.js/issues/307)) (73f3481 (https://github.com/ERC725Alliance/erc725.js/commit/73f34818fe152c3ab5299177adc0eddfed6886
- new encoding for static value types (not arrays []) (#288 (https://github.com/ERC725Alliance/erc725.js/issues/288)) (f0b04da (https://github.com/ERC725Alliance/erc725.js/commit/f0b04daa57a281c537a8f28594439573188f0d

0.18.0 (https://github.com/ERC725Alliance/erc725.js/compare/v0.17.2...v0.18.0) (2023-08-03)

⚠ BREAKING CHANGES

- new encoding for static value types (not arrays []) (#288)
- change ArrayLength value from uint256 to uint128 (#287)

Features

- add checkPermissions function (17d2258 (https://github.com/ERC725Alliance/erc725.js/commit/17d225843c236951ef1515a0ff91095b5ef27cc
- Add new feature "Decode Mapping Key" (8c1f1fc (https://github.com/ERC725Alliance/erc725.js/commit/8c1f1fcfb15fa43d1d3934b0b15f09d47902bb4

Bug Fixes

- decode any uint256 as number not string (#289 (https://github.com/ERC725Alliance/erc725.js/issues/289)) (37203f1 (https://github.com/ERC725Alliance/erc725.js/commit/37203f14d313a0caff75724dc74175c741c1b5
- variable naming (44b4785 (https://github.com/ERC725Alliance/erc725.js/commit/44b47851ed63b817edc21c63655d67bac13a7

improvement

- change ArrayLength value from uint256 to uint128 (#287 (https://github.com/ERC725Alliance/erc725.js/issues/287)) (c95ee8a (https://github.com/ERC725Alliance/erc725.js/commit/c95ee8a53bf25bcf47777054af27cae1fbad8b2
- new encoding for static value types (not arrays []) (#288 (https://github.com/ERC725Alliance/erc725.js/issues/288)) (f0b04da (https://github.com/ERC725Alliance/erc725.js/commit/f0b04daa57a281c537a8f28594439573188f0d

0.17.2 (https://github.com/ERC725Alliance/erc725.js/compare/v0.17.1...v0.17.2) (2023-03-14)

- removed ERC725JSONSchemaKeyType duplicate value (060ee6c

(https://github.com/ERC725Alliance/erc725.js/commit/060ee6ce23bda328f727140419de7590f48fc3
- wrong web3 import (337269e (https://github.com/ERC725Alliance/erc725.js/commit/337269eb22f82e6f44f9b8c9be4840fa6cd676e

0.17.1 (https://github.com/ERC725Alliance/erc725.js/compare/v0.17.0...v0.17.1) (2023-02-08)

⚠ BREAKING CHANGES

- update schemas (#274)

0.17.0 (https://github.com/ERC725Alliance/erc725.js/compare/v0.16.0...v0.17.0) (2023-02-07)

⚠ BREAKING CHANGES

- refactor: change permission ADDPERMISSIONS -> ADDCONTROLLER (122efa9 (https://github.com/ERC725Alliance/erc725.js/commit/122efa99b4858a97cd3b1d6f5c07d85983e036

Features

- add encoding / decoding for bytes[CompactBytesArray] (#261 (https://github.com/ERC725Alliance/erc725.js/issues/261)) (8d3e4e9 (https://github.com/ERC725Alliance/erc725.js/commit/8d3e4e994957dbf1f54eef4572b072bc308c61
- add decoding/encoding support for tuples of CompactBytesArray (#264 (https://github.com/ERC725Alliance/erc725.js/issues/264)) (d9ce0f0 (https://github.com/ERC725Alliance/erc725.js/commit/d9ce0f08b61a4a04ebc98c6b996513fb432768
- add encoding/decoding for other types of compactBytesArray (#262 (https://github.com/ERC725Alliance/erc725.js/issues/262)) (9268a32 (https://github.com/ERC725Alliance/erc725.js/commit/9268a3205b54f81069e6217c827dff69d41668
- Add support for bool (valueType) and Boolean (valueContent) (#266 (https://github.com/ERC725Alliance/erc725.js/issues/266)) (86d606e (https://github.com/ERC725Alliance/erc725.js/commit/86d606e677a69eebaaaeb9467c37e5f6303efb

Bug Fixes

- Encode key name should parse any number (hex or decimal) for uint type (eb7385e (https://github.com/ERC725Alliance/erc725.js/commit/eb7385e6e97f6f069c4be5331f0c4547b79faab

0.16.0 (https://github.com/ERC725Alliance/erc725.js/compare/v0.15.0...v0.16.0) (2022-10-27)

⚠ BREAKING CHANGES

- It is now recommended to initialise the library with an RPC URL over the Web3 or Ethers provider, you can check the erc725.js documentation (https://docs.lukso.tech/tools/erc725js/providers#rpc-url) to get more information.

Features

- add supportsInterface (https://docs.lukso.tech/tools/erc725js/classes/ERC725#supportsinterface) (#243 (https://github.com/ERC725Alliance/erc725.js/issues/243)) (a2b0828 (https://github.com/ERC725Alliance/erc725.js/commit/a2b08288e9aaede8a1a3307c1371672eb7b5(
- make library compatible with RPC urls (263de19 (https://github.com/ERC725Alliance/erc725.js/commit/263de1983f08c9f31f0cc931e581fe8af52bd54

Bug Fixes

- change lsp7 interface id from 0xe33f65c3 to 0x5fcaac27 (6aa6eb3 (https://github.com/ERC725Alliance/erc725.js/commit/6aa6eb30d427609c89ebc1920fc9ecf03c0dd6
- update return type for fetchData (#247 (https://github.com/ERC725Alliance/erc725.js/issues/247)) (7ffcd64 (https://github.com/ERC725Alliance/erc725.js/commit/7ffcd64b2c0ee841b71472d0d6d869e4149db3

0.15.0 (https://github.com/ERC725Alliance/erc725.js/compare/v0.14.4...v0.15.0) (2022-09-14)

⚠ BREAKING CHANGES

- add LSP6 ENCRYPT permission in for encrypt/decrypt permissions methods. (#223)

0.14.4 (https://github.com/ERC725Alliance/erc725.js/compare/v0.14.3...v0.14.4) (2022-07-25)

Features

- expose dynamic parts to public encodeKeyName (#213 (https://github.com/ERC725Alliance/erc725.js/issues/213)) (73f1265 (https://github.com/ERC725Alliance/erc725.js/commit/73f126570eaf5f118c48859ee878608afc48a0a

0.14.3 (https://github.com/ERC725Alliance/erc725.js/compare/v0.14.2...v0.14.3) (2022-07-07)

Features

- add BitArray in valueContent encoding map (3498502 (https://github.com/ERC725Alliance/erc725.js/commit/3498502da0b531559e557fc4a9e6c2851b480

Bug Fixes

- LSP4Creators[] valueType (6ddbf47 (https://github.com/ERC725Alliance/erc725.js/commit/6ddbf473039b17af562d56f1f12be5194a06047

0.14.2 (https://github.com/ERC725Alliance/erc725.js/compare/v0.14.1...v0.14.2) (2022-06-24)

Bug Fixes

- Update JsonRpc eth_call parameters (470e846 (https://github.com/ERC725Alliance/erc725.js/commit/470e8461e581f0763fe3fddc8f604cc55002f7a

0.14.1 (https://github.com/ERC725Alliance/erc725.js/compare/v0.14.0...v0.14.1) (2022-06-15)

Minor update to update the LSP-2 schemas.

0.14.0 (https://github.com/ERC725Alliance/erc725.js/compare/v0.13.0...v0.14.0) (2022-06-14)

⚠ BREAKING CHANGES

- fetchData as same output as decodeData
- getData as same output as decodeData
- add dynamic keys for getData
- use array for decodeData
- use array for encodeData

Features

- add dynamic keys for getData (7a46786 (https://github.com/ERC725Alliance/erc725.js/commit/7a46786105d35ca0b33ce5158be98866f3fbfd
- add dynamicKeys for decodeData (f386e15 (https://github.com/ERC725Alliance/erc725.js/commit/f386e15396cbf6c2b842302f1c85b41a0798b6
- add non array input on decodeData (0774a86 (https://github.com/ERC725Alliance/erc725.js/commit/0774a86d652e205814863d07319adef2266f6
- add support for hashed key for encodeData (23323a0 (https://github.com/ERC725Alliance/erc725.js/commit/23323a02a6bfb81be1eba24207fe3539423678
- add BytesN value content (#184 (https://github.com/ERC725Alliance/erc725.js/issues/184)) (7e073e4 (https://github.com/ERC725Alliance/erc725.js/commit/7e073e4dfc7094aac55935fffc18ef14d16a24c8
- add tuples support (7f3d1a0 (https://github.com/ERC725Alliance/erc725.js/commit/7f3d1a09c3e6058b76a5b3ceca8bc1f454e634

Bug Fixes

- encodeKeyName returns lowercase keys (80566eb (https://github.com/ERC725Alliance/erc725.js/commit/80566eb2358db0ff90e42028ee5b1b5bca206b

improvement

- fetchData as same output as decodeData (59c3a87 (https://github.com/ERC725Alliance/erc725.js/commit/59c3a879fefb2b9bfe46b9bea91ff6bd2a528df1
- getData as same output as decodeData (0f3b149 (https://github.com/ERC725Alliance/erc725.js/commit/0f3b149f2280e6025a05e8e9ed306facfa63601
- use array for decodeData (261d100 (https://github.com/ERC725Alliance/erc725.js/commit/261d1007f4ff63abd9d794f4e64e5b408ce7c1a
- use array for encodeData (a2e6cdd (https://github.com/ERC725Alliance/erc725.js/commit/a2e6cdd5cca778f9015c71a624cc3953e2e0fd

0.13.0 (https://github.com/ERC725Alliance/erc725.js/compare/v0.12.0...v0.13.0) (2022-06-02)

⚠ BREAKING CHANGES

- remove old LSP2 Key Types (Bytes20..)

Features

- add dynamic keys for encodeKeyName (#168 (https://github.com/ERC725Alliance/erc725.js/issues/168)) (fc614b0 (https://github.com/ERC725Alliance/erc725.js/commit/fc614b0b92d3b5c7b86586f4be7e3200ea968

improvement

- change fetchData output to non object (1d4d570 (https://github.com/ERC725Alliance/erc725.js/commit/1d4d57077a7766b3490477efb20f194fc4e00d
- remove old LSP2 Key Types (Bytes20..) (1e1cd43 (https://github.com/ERC725Alliance/erc725.js/commit/1e1cd43e7693db5f12200aef6e282fa14e655e

0.12.0 (https://github.com/ERC725Alliance/erc725.js/compare/v0.11.1...v0.12.0) (2022-05-19)

⚠ BREAKING CHANGES

- change encodeData result structure

improvement

- change encodeData result structure (10da619 (https://github.com/ERC725Alliance/erc725.js/commit/10da619bf3eee18f6a764e0c8af8c36d9caf8d5

0.11.1 (https://github.com/ERC725Alliance/erc725.js/compare/v0.11.0...v0.11.1) (2022-04-06)

This version fix the npm pack error.

Bug Fixes

- do not load wrong schemas (66dc3e6 (https://github.com/ERC725Alliance/erc725.js/commit/66dc3e648ad1a9aeabe66e5ae2aeb15cf3f747

0.11.0 (https://github.com/ERC725Alliance/erc725.js/compare/v0.10.0...v0.11.0) (2022-04-05)

⚠ BREAKING CHANGES

- the output of getData is not an object anymore, but the value directly if the input is a string.
- if fetchData is called with a string, the output will be the value itself, not an object anymore.

Features

- add schemas at the root and improve docs (#121 (https://github.com/ERC725Alliance/erc725.js/issues/121)) (e37fb39 (https://github.com/ERC725Alliance/erc725.js/commit/e37fb3926bcb682df00c632feb3b3a8b1700d2
- change the output of getData for string input (3592c1b (https://github.com/ERC725Alliance/erc725.js/commit/3592c1bb335e9a1bce824bc3ef8667e98ae9e8

0.10.0 (https://github.com/ERC725Alliance/erc725.js/compare/v0.9.2...v0.10.0) (2022-03-11)

Features

- add isValidSignature (6490751 (https://github.com/ERC725Alliance/erc725.js/commit/6490751c009e435ac23eb98e6bfe64d271b85

0.9.2 (https://github.com/ERC725Alliance/erc725.js/compare/v0.9.1...v0.9.2) (2022-02-21)

Features

- add LSP1 schema (4f849da (https://github.com/ERC725Alliance/erc725.js/commit/4f849dac01116e6f019e04fea950b42d227191

Bug Fixes

- update ERC725Y JSON Schemas to latest LSPs specs (#92 (https://github.com/ERC725Alliance/erc725.js/issues/92)) (3485baa (https://github.com/ERC725Alliance/erc725.js/commit/3485baa347cf9a194bd0c4ea2a1e8c61922b6
- wait until a promise is resolved when using ethereum provider (5efe641 (https://github.com/ERC725Alliance/erc725.js/commit/5efe6414b6e4a4250d8c402baa887a269e6f8

0.9.1 (https://github.com/ERC725Alliance/erc725.js/compare/v0.9.0...v0.9.1) (2022-02-01)

Bug Fixes

- getSchema array (#95 (https://github.com/ERC725Alliance/erc725.js/issues/95)) (8ce5ff1 (https://github.com/ERC725Alliance/erc725.js/commit/8ce5ff1c81ece3534fd557d978bda4107dfd380

0.9.0 (https://github.com/ERC725Alliance/erc725.js/compare/v0.8.0...v0.9.0) (2022-01-06)

⚠ BREAKING CHANGES

- GraphQL / Apollo has been removed

Features

- add encodeKeyName method (#86 (https://github.com/ERC725Alliance/erc725.js/issues/86)) (7cf43ba (https://github.com/ERC725Alliance/erc725.js/commit/7cf43babbf461a05636d31941237adf94a3d36
- add getSchema (#85 (https://github.com/ERC725Alliance/erc725.js/issues/85)) (7f677d0 (https://github.com/ERC725Alliance/erc725.js/commit/7f677d0b6b08061773a151d2e91a21156ca59
- LSP6 Permissions encoding methods (#84 (https://github.com/ERC725Alliance/erc725.js/issues/84)) (2e1031a (https://github.com/ERC725Alliance/erc725.js/commit/2e1031a047f19b2fc98104b7df58eecb1424b6

Bug Fixes

- empty JSON url return null instead of crash (#61 (https://github.com/ERC725Alliance/erc725.js/issues/61)) (2d1e417 (https://github.com/ERC725Alliance/erc725.js/commit/2d1e417facbc9b2c5b1f4fae62d46b498f3f760
- remove GraphQL support (#83 (https://github.com/ERC725Alliance/erc725.js/issues/83)) (a0a5e93 (https://github.com/ERC725Alliance/erc725.js/commit/a0a5e93bff3e4a5cc759c7b8662f7df523fa484

0.8.0 (https://github.com/ERC725Alliance/erc725.js/compare/v0.6.2-beta.4...v0.8.0) (2021-11-22)

⚠ BREAKING CHANGES

- remove deprecated elementValueType / elementValueContent keys (#45)

Features

- add support for new getData([]) (#48 (https://github.com/ERC725Alliance/erc725.js/issues/48)) (6cbb1e7 (https://github.com/ERC725Alliance/erc725.js/commit/6cbb1e76e3df8b862ee35e436aaddea24f86e2
- remove deprecated elementValueType / elementValueContent keys (#45 (https://github.com/ERC725Alliance/erc725.js/issues/45)) (a326cd6 (https://github.com/ERC725Alliance/erc725.js/commit/a326cd6560a8a9de6c68db61c919c07f4f71e3

Bug Fixes

- handling of missing keys (#58 (https://github.com/ERC725Alliance/erc725.js/issues/58)) (9431f85 (https://github.com/ERC725Alliance/erc725.js/commit/9431f85dd62785305f5b32ca6c4a4e9e3c9b78
- wrong return type for fetchData (#54 (https://github.com/ERC725Alliance/erc725.js/issues/54)) (0ce147a

(https://github.com/ERC725Alliance/erc725.js/commit/0ce147ac0774a3ec0b404896da02b923bbfbd(

0.7.0 (https://github.com/ERC725Alliance/erc725.js/compare/v0.6.2-beta.4...v0.7.0) (2021-11-02)

⚠ BREAKING CHANGES

- remove deprecated elementValueType / elementValueContent keys (#45)

Features

- add support for new getData([]) (#48 (https://github.com/ERC725Alliance/erc725.js/issues/48)) (6cbb1e7 (https://github.com/ERC725Alliance/erc725.js/commit/6cbb1e76e3df8b862ee35e436aaddea24f86e2
- remove deprecated elementValueType / elementValueContent keys (#45 (https://github.com/ERC725Alliance/erc725.js/issues/45)) (a326cd6 (https://github.com/ERC725Alliance/erc725.js/commit/a326cd6560a8a9de6c68db61c919c07f4f71e3

0.6.1 (https://github.com/ERC725Alliance/erc725.js/compare/v0.6.0...v0.6.1) (2021-08-13)

Docs

- docs: Update missed occurrences of "erc725.js"

0.6.0 (https://github.com/ERC725Alliance/erc725.js/compare/v0.5.7...v0.6.0) (2021-08-13)

Feature

- API: Adjusted API to be more consistent
    - https://github.com/ERC725Alliance/erc725.js/issues/30
    - https://github.com/ERC725Alliance/erc725.js/pull/31

Bug Fixes

- mocha: increase timeout time (b7ce1a0 (https://github.com/ERC725Alliance/erc725.js/commit/b7ce1a07711b8251f4447d613c4c5a522b5e2(

0.5.7 (https://github.com/ERC725Alliance/erc725.js/compare/v0.2.0...v0.5.7) (2021-07-30)

Bug Fixes

- publish: ensure clean build folder (973e09b (https://github.com/ERC725Alliance/erc725.js/commit/973e09b936277c254fdc9c15d4d5d89fc4dc05
</file>

<file>

path: /CONTRIBUTING.md

url: https://github.com/ERC725Alliance/erc725.js/blob/main/CONTRIBUTING.md

Contributing to erc725.js

Commits and PRs

This project uses Conventional Commits to generate release notes and to determine versioning. Commit messages should adhere to this standard and be of the form:

```
git commit -m "feat: Add new feature x"
git commit -m "fix: Fix bug in feature x"
git commit -m "docs: Add documentation for feature x"
git commit -m "test: Add test suite for feature x"
```

Further details on conventional commits can be found here: https://www.conventionalcommits.org/en/v1.0.0/

Building

```
npm run build
```

This will build the library into /build

Testing

```
npm test
```

Will build and then publish the package to npm.

Release

To release and publish a new version, check RELEASE (./RELEASE.md).
</file>

<file>

path: /README.md

url: https://github.com/ERC725Alliance/erc725.js/blob/main/README.md

erc725.js · license Apache (./LICENSE) npm v0.21.2 (https://www.npmjs.com/package/@erc725/erc725.js)

<h2 align="center">@erc725/erc725.js</h2>
Allows for interfacing with ERC725Y compliant contracts on an EVM blockchain.

For more information see Documentation.

Installation

:warning: *This package is currently in early stages of development,
 use for testing or experimentation purposes only.*

```
npm install @erc725/erc725.js
```

```
import { ERC725 } from '@erc725/erc725.js';
// Or alternatively the default export
import ERC725 from '@erc725/erc725.js';
```

If you install it on the backend side, you may need to also install [isomorphic-fetch (https://www.npmjs.com/package/isomorphic-fetch)](https://www.npmjs.com/package/isomorphic-fetch).

Usage

```
import { ERC725 } from '@erc725/erc725.js';
import Web3 from 'web3';

// Part of LSP3-Profile Schema
// https://github.com/lukso-network/LIPs/blob/master/LSPs/LSP-3-Profile-Metadata.md
const schema = [
  {
    name: 'SupportedStandards:LSP3Profile',
    key: '0xeafec4d89fa9619884b600005ef83ad9559033e6e941db7d7c495acdce616347',
    keyType: 'Mapping',
    valueContent: '0x5ef83ad9',
    valueType: 'bytes',
  },
  {
    name: 'LSP3Profile',
    key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
    keyType: 'Singleton',
    valueContent: 'JSONURL',
    valueType: 'bytes',
  },
  {
    name: 'LSP1UniversalReceiverDelegate',
    key: '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
    keyType: 'Singleton',
    valueContent: 'Address',
    valueType: 'address',
  },
];

const address = '0x3000783905Cc7170cCCe49a4112Deda952DDBe24';
const RPC_URL = 'https://rpc.testnet.lukso.network';
const config = {
  ipfsGateway: 'https://2eff.lukso.dev/ipfs/',
};

const myErc725 = new ERC725(schema, address, RPC_URL, config);
```

Usage

```
await myErc725.getOwner();
// > '0x28D25E70819140daF65b724158D00c373D1a18ee'

await myErc725.getData();
/**
[
  {
    name: 'SupportedStandards:LSP3Profile',
    key: '0xeafec4d89fa9619884b600005ef83ad9559033e6e941db7d7c495acdce616347',
    value: '0x5ef83ad9',
  },
  {
```

```
  name: 'LSP1UniversalReceiverDelegate',
  key: '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
  value: '0x50A02EF693fF6961A7F9178d1e53CC8BbE1DaD68',
 },
 {
  name: 'LSP12IssuedAssets[]',
  key: '0x7c8c3416d6cda87cd42c71ea1843df28ac4850354f988d55ee2eaa47b6dc05cd',
  value: [
   '0xc444009d38d3046bb0cF81Fa2Cd295ce46A67C78',
   '0x4fEbC3491230571F6e1829E46602e3b110215A2E',
   '0xB92a8DdA288638491AEE5C2a003D4CAbfa47aE3F',
   '0x1e52e7F1707dcda57dD33F003B2311652A465acA',
   '0x0BDA71aA980D37Ea56E8a3784E4c309101DAf3E4',
   '0xfDB4D9C299438B9839e9d04E34B9609C5b56600D',
   '0x081D3F0bff8ae2339cb65113822eEc1510704d5c',
   '0x55C98c6944B7497FaAf4db0386a1aD1E6efF526E',
   '0x90D1a1D68fa23AEEE991220703f1a1C3782e0b35',
   '0xdB5AB19792d9fB61c1Dff57810Fb7C6f839Af8ED'
  ],
 },
 {
  name: 'LSP3Profile',
  key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
  value: {
   verification: {
    method: 'keccak256(utf8)',
    data: '0x70546a2accab18748420b63c63b5af4cf710848ae83afc0c51dd8ad17fb5e8b3'
   },
   url: 'ipfs://QmecrGejUQVXpW4zS948pNvcnQrJ1KiAoM6bdfrVcWZsn5',
  },
 },
]
*/

await myErc725.fetchData();
/**
[
 {
  name: 'SupportedStandards:LSP3Profile',
  key: '0xeafec4d89fa9619884b600005ef83ad9559033e6e941db7d7c495acdce616347',
  value: '0x5ef83ad9'
 },
 {
  name: 'LSP3Profile',
  key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
  value: { LSP3Profile: [Object] }
 },
 {
  name: 'LSP1UniversalReceiverDelegate',
  key: '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
  value: '0xE2D6038acD92200790Df695Ebd13856CdF2a6942'
 },
```

```
{
  name: 'LSP12IssuedAssets[]',
  key: '0x7c8c3416d6cda87cd42c71ea1843df28ac4850354f988d55ee2eaa47b6dc05cd',
  value: [
    '0xc444009d38d3046bb0cF81Fa2Cd295ce46A67C78',
    '0x4fEbC3491230571F6e1829E46602e3b110215A2E',
    '0xB92a8DdA288638491AEE5C2a003D4CAbfa47aE3F',
    '0x1e52e7F1707dcda57dD33F003B2311652A465acA',
    '0x0BDA71aA980D37Ea56E8a3784E4c309101DAf3E4',
    '0xfDB4D9C299438B9839e9d04E34B9609C5b56600D',
    '0x081D3F0bff8ae2339cb65113822eEc1510704d5c',
    '0x55C98c6944B7497FaAf4db0386a1aD1E6efF526E',
    '0x90D1a1D68fa23AEEE991220703f1a1C3782e0b35',
    '0xdB5AB19792d9fB61c1Dff57810Fb7C6f839Af8ED'
  ]
 }
]
*/
```

Contributing

Please check CONTRIBUTING (./CONTRIBUTING.md).

License

erc725.js is Apache 2.0 licensed (./LICENSE).
</file>

<file>

path: /RELEASE.md

url: https://github.com/ERC725Alliance/erc725.js/blob/main/RELEASE.md

Release Process

Releases are published to npm using Release Please (https://github.com/googleapis/release-please).

This package automates CHANGELOG generation, version bumps and npm releases by parsing the git history, looking for Conventional Commit messages (https://www.conventionalcommits.org/).

When changes and feature PRs are merged from develop to main, release-please will open and maintain a release PR with the updated CHANGELOG and new version number. When this PR is merged, a release will be created and the package published to NPM.

1. Merge develop into main.
2. Release Please will create the release PR going to main.
3. Merge the generated release PR.
4. Package will be published to NPM.

Conventional Commit prefixes?

Commits should follow the Conventional Commit messages standard (https://www.conventionalcommits.org/).

The following commit prefixes will result in changes in the CHANGELOG:

- fix: which represents bug fixes, and correlates to a SemVer (https://semver.org/) patch.
- feat: which represents a new feature, and correlates to a minor version increase.
(indicated by the !) and will result in a SemVer major version increase.
- feat!:, or fix!:, refactor!:, etc., which represent a breaking change
- build: Changes that affect the build system or external dependencies.
- ci: Changes to our CI configuration files and scripts.
- docs: Documentation only changes.
- perf: A code change that improves performance.
- refactor: A code change that neither fixes a bug nor adds a feature.
- style: Changes that do not affect the meaning of the code (white-space, formatting, missing semi-colons, etc).
- test: Adding missing tests or correcting existing tests.
- chore: Other

Release with a custom version number

When a commit to the main branch has Release-As: x.x.x (case insensitive) in the commit body, Release Please will open a new pull request for the specified version.

git commit --allow-empty -m "chore: release 2.0.0" -m "Release-As: 2.0.0" results in the following commit message:

```
chore: release 2.0.0

Release-As: 2.0.0
```

How can I fix release notes?

If you have merged a pull request and would like to amend the commit message used to generate the release notes for that commit, you can edit the body of the merged pull requests and add a section like:

```
BEGIN_COMMIT_OVERRIDE
feat: add ability to override merged commit message

fix: another message
chore: a third message
END_COMMIT_OVERRIDE
```

The next time Release Please runs, it will use that override section as the commit message instead of the merged commit message.
</file>

<file>

path: /docs/README.md

url: https://github.com/ERC725Alliance/erc725.js/blob/main/docs/README.md

erc725.js docs

The markdown files in this folder are published on the LUKSO docs portal:
https://docs.lukso.tech/tools/erc725js/getting-started (https://docs.lukso.tech/tools/erc725js/getting-started)
</file>

<file>

path: /docs/classes/ERC725.md

url: https://github.com/ERC725Alliance/erc725.js/blob/main/docs/classes/ERC725.md

---

sidebar_position: 1

ERC725

checkPermissions

```
myErc725.checkPermissions(requiredPermissions, grantedPermissions);
```

```
ERC725.checkPermissions(requiredPermissions, grantedPermissions);
```

Check if the required permissions are included in the granted permissions as defined by the LSP6 KeyManager Standard (https://docs.lukso.tech/standards/universal-profile/lsp6-key-manager).

:::info

checkPermissions is available as either a static or non-static method so can be called without instantiating an ERC725 object.

:::

Parameters

1. requiredPermissions - String[] | String

An array of required permissions or a single required permission. (32bytes hex or the official name of the permission).

2. grantedPermissions - String

The granted permissions. (32bytes hex).

Returns

Type	Description
boolean	A boolean value indicating whether the required permissions are included in the granted permissions as defined by the LSP6 KeyManager Standard

(https://docs.lukso.tech/standards/universal-profile/lsp6-key-manager)

Permission-Name Example

```
const requiredPermissions = 'CHANGEOWNER';
const grantedPermissions =
  '0x000000000000000000000000000000000000000000000000000000000000ff51';
ERC725.checkPermissions(requiredPermissions, grantedPermissions);
// true

// This method is also available on the instance:

const requiredPermissions = ['CHANGEOWNER', 'CALL'];
const grantedPermissions =
  '0x0000000000000000000000000000000000000000000000000000000000000051';
myErc725.checkPermissions(requiredPermissions, grantedPermissions);
// false
```

32bytes hex Example

```
const requiredPermissions = [
  '0x0000000000000000000000000000000000000000000000000000000000000001',
  '0x0000000000000000000000000000000000000000000000000000000000000800',
];
const grantedPermissions =
  '0x0000000000000000000000000000000000000000000000000000000000000051';

ERC725.checkPermissions(requiredPermissions, grantedPermissions);
// false

// This method is also available on the instance:

const requiredPermissions =
  '0x0000000000000000000000000000000000000000000000000000000000000001';
const grantedPermissions =
  '0x0000000000000000000000000000000000000000000000000000000000000051';

myErc725.checkPermissions(requiredPermissions, grantedPermissions);
// true
```

---

decodeData

```
myErc725.decodeData(data [, schemas]);
```

```
ERC725.decodeData(data, schemas);
```

If you are reading the key-value store from an ERC725 smart-contract you can use the decodeData function to do the decoding for you.

:::tip
If you want total convenience, it is recommended to use the fetchData (ERC725.md#fetchdata) function, which automatically decodes and fetches external references.
:::

Parameters

1. data - Object or array of Objects

An object or array of objects containing the following properties:

Name	Type	Description
keyName	string	Can be either the named key (i.e. LSP3Profile, LSP12IssuedAssetsMap:<address>) or the hashed key (with or without 0x prefix, i.e. 0x5ef... or 5ef...).
dynamicKeyParts (optional)	string or string[ ]	If keyName is a dynamic key, the dynamic parts of the keyName that will be used for encoding the key.
value	string or string[ ]	The value that should be decoded. Can be a string, an array of string or a JSON...

The keyName also supports dynamic keys for Mapping (https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-2-ERC725YJSONSchema.md#mapping) and MappingWithGrouping (https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-2-ERC725YJSONSchema.md#mapping). Therefore, you can use variables in the key name such as LSP12IssuedAssetsMap:<address>. In that case, the value should also set the dynamicKeyParts property:

- dynamicKeyParts: string or string[ ] which holds the variables that needs to be encoded.

2. schemas - Array of Objects (optional)

An array of extra LSP-2 ERC725YJSONSchema (https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-2-ERC725YJSONSchema.md) objects that can be used to find the schema. If called on an instance, it is optional and it will be concatenated with the schema provided on instantiation.

Returns

Name	Type	Description
decodedData	Object or Array	The decoded data as defined and expected in the following schemas.

:::info

- If the input is an array of objects, the values will be returned in an array.
- If the input is a single object, the output will be the decodedData object directly.

:::

Single-Key Example

```
myErc725.decodeData([
  {
    keyName: 'LSP3Profile',
    value:

'0x6f357c6a820464ddfac1bec070cc14a8daf04129871d458f2ca94368aae8391311af6361696670733a2f2f516d597231564a4c776572673670456f736364685647756 76f33397061367279634 55a4c6a745250446657383 4554178',
  },
]);
/**
[
  {
    name: 'LSP3Profile',
    key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
    value: {
      verification: {
        method: 'keccak256(utf8)',
        data: '0x820464ddfac1bec070cc14a8daf04129871d458f2ca94368aae8391311af6361'
      },
      url: 'ipfs://QmYr1VJLwerg6pEoscdhVGugo39pa6rycEZLjtRPDfW84UAx',
    },
  },
]
*/

myErc725.decodeData({
  keyName: 'LSP3Profile',
  value:

'0x6f357c6a820464ddfac1bec070cc14a8daf04129871d458f2ca94368aae8391311af6361696670733a2f2f516d597231564a4c776572673670456f736364685647756 76f33397061367279634 55a4c6a745250446657383 4554178',
});
/**
{
  name: 'LSP3Profile',
  key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
  value: {
    verification: {
      method: 'keccak256(utf8)',
      data: '0x820464ddfac1bec070cc14a8daf04129871d458f2ca94368aae8391311af6361'
    },
    url: 'ipfs://QmYr1VJLwerg6pEoscdhVGugo39pa6rycEZLjtRPDfW84UAx',
  },
}
*/
```

Multi-Key Example

```
myErc725.decodeData([
  {
    keyName: 'LSP3Profile',
    value:

'0x6f357c6a820464ddfac1bec070cc14a8daf04129871d458f2ca94368aae8391311af6361696670733a2f2f516d597231564a4c776572673670456f736364685647756 76f33397061367279634 55a4c6a745250446657383 4554178',
  },
  {
    keyName: 'LSP12IssuedAssets[]',
    value: [
      {
        key: '0x7c8c3416d6cda87cd42c71ea1843df28ac4850354f988d55ee2eaa47b6dc05cd',
        value: '0x00000000000000000000000000000002',
      },
      {
        key: '0x7c8c3416d6cda87cd42c71ea1843df2800000000000000000000000000000000',
        value: '0xd94353d9b005b3c0a9da169b768a31c57844e490',
      },
      {
        key: '0x7c8c3416d6cda87cd42c71ea1843df2800000000000000000000000000000001',
        value: '0xdaea594e385fc724449e3118b2db7e86dfba1826',
      },
    ],
  },
]);
/**
[
  {
    name: 'LSP3Profile',
    key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
    value: {
      verification: {
        data: '0x820464ddfac1bec070cc14a8daf04129871d458f2ca94368aae8391311af6361',
        method: 'keccak256(utf8)'
      },
      url: 'ipfs://QmYr1VJLwerg6pEoscdhVGugo39pa6rycEZLjtRPDfW84UAx',
    },
  },
  {
    name: 'LSP12IssuedAssets[]',
    key: '0x7c8c3416d6cda87cd42c71ea1843df28ac4850354f988d55ee2eaa47b6dc05cd',
    value: [
      '0xD94353D9B005B3c0A9Da169b768a31C57844e490',
      '0xDaea594E385Fc724449E3118B2Db7E86dFBa1826',
    ],
  },
];
*/
```

Dynamic-Key Example

```
const schemas = [
  {
    name: 'MyKeyName:<bytes32>:<bool>',
    key: '0x...',
    keyType: 'Singleton',
    valueType: 'bytes',
    valueContent: 'JSONURL',
  },
];

myErc725.decodeData(
  [
    {
      keyName: 'MyKeyName:<bytes32>:<bool>',
      dynamicKeyParts: [
        '0xaaaabbbbccccddddeeeeffff11112222333344445555666677778888999aaaa',
        'true',
      ],
      value:

'0x6f357c6a820464ddfac1bec070cc14a8daf04129871d458f2ca94368aae8391311af6361696670733a2f2f516d597231564a4c776572673670456f736364685647756 76f33397061367279634 55a4c6a745250446657383 4554178',
    },
  ],
  schemas,
);
/**
[
  {
    name: 'MyKeyName:aaaabbbbccccddddeeeeffff11112222333344445555666677778888999aaaa:true',
    key: '0x35e6950bc8d2aaaabbbb0000000000000000000000000000000000000000000001',
    value: {
      verification: {
        method: 'keccak256(utf8)',
        data: '0x820464ddfac1bec070cc14a8daf04129871d458f2ca94368aae8391311af6361'
      },
      url: 'ifps://QmYr1VJLwerg6pEoscdhVGugo39pa6rycEZLjtRPDfW84UAx',
    },
  },
];
*/
```

decodePermissions

```
ERC725.decodePermissions(permission);
```

Decodes permissions from hexadecimal defined by the [LSP6 KeyManager Standard (https://docs.lukso.tech/standards/universal-profile/lsp6-key-manager)](https://docs.lukso.tech/standards/universal-profile/lsp6-key-manager).

:::info

decodePermissions is available as either a static or non-static method so can be called without instantiating an ERC725 object.

:::

Parameters

1. permission - String

The encoded permission (32bytes hex).

Returns

Name	Type	Description
decodedPermissions	Object	An object specifying whether default LSP6 permissions are included in provided hexademical string.

Example

```
ERC725.decodePermissions('0x0000000000000000000000000000000000000000000000000000000000000110'),
/**
{
  CHANGEOWNER: false,
  EDITPERMISSIONS: false,
  ADDCONTROLLER: false,
  SETDATA: false,
  CALL: true,
  STATICCALL: false,
  DELEGATECALL: false,
  DEPLOY: false,
  TRANSFERVALUE: true,
  SIGN: false,
}
*/

ERC725.decodePermissions('0x000000000000000000000000000000000000000000000000000000000000000a'),
/**
{
  CHANGEOWNER: false,
  EDITPERMISSIONS: true,
  ADDCONTROLLER: false,
  SETDATA: true,
  CALL: false,
  STATICCALL: false,
  DELEGATECALL: false,
  DEPLOY: false,
  TRANSFERVALUE: false,
  SIGN: false,
}
*/

// This method is also available on the instance:
myErc725.decodePermissions('0x0000000000000000000000000000000000000000000000000000000000000110'),
```

encodeData

```
myErc725.encodeData(data [, schemas]);
```

```
ERC725.encodeData(data, schemas);
```

Encode the data of a smart contract according to your ERC725JSONSchema so that you can store the information in smart contracts.

:::tip

When encoding JSON, it is possible to pass in the JSON object and the URL where it is available publicly. The JSON will be hashed with keccak256.

:::

:::info

When encoding some values using specific string or bytesN as valueType, if the data passed is a non-hex value, *erc725.js* will convert the value
to its utf8-hex representation for you. For instance:

- If valueType is string and you provide a number as input.

Example: input 42 --> will encode as 0x3432 (utf-8 hex code for 4 = 0x34, for 2 = 0x32).

- If valueType is bytes32 or bytes4, it will convert as follow:

Example 1: input week encoded as bytes4 --> will encode as 0x7765656b.

Example 2: input 1122334455 encoded as bytes4 --> will encode as 0x42e576f7.

:::

Parameters

1. data - Array of Objects

An array of objects containing the following properties:

Name	Type	Description
keyName	string	Can be either the named key (i.e. LSP3Profile, LSP12IssuedAssetsMap:<address>) or the hashed key (with or without 0x prefix, i.e. 0x5ef... or 5ef...).
dynamicKeyParts (optional)	string or string[ ]	The dynamic parts of the keyName that will be used for encoding the key.
value	string or string[ ] JSON todo	The value that should be encoded. Can be a string, an array of string or a JSON...

The keyName also supports dynamic keys for Mapping (https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-2-ERC725YJSONSchema.md#mapping) and MappingWithGrouping (https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-2-ERC725YJSONSchema.md#mapping). Therefore, you can use variables in the key name such as LSP12IssuedAssetsMap:<address>. In that case, the value should also set the dynamicKeyParts property:

- dynamicKeyParts: string or string[ ] which holds the variables that needs to be encoded.

2. schemas - Array of Objects (optional)

An array of extra LSP-2 ERC725YJSONSchema (https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-2-ERC725YJSONSchema.md) objects that can be used to find the schema. If called on an instance, it is optional and it will be concatenated with the schema provided on instantiation.

Returns

Name	Type	Description

encodedData	Object	An object containing the encoded keys and values according to the [LSP2 ERC725Y JSON Schema (https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-2-ERC725YJSONSchema.md)](https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-2-ERC725YJSONSchema.md) of the data which was passed

After the data is encoded, the object is ready to be stored in smart contracts.

Examples

<details>
<summary>Encode a <code>JSONURL</code> with JSON and uploaded URL</summary>

```
myErc725.encodeData([
  {
    keyName: 'LSP3Profile',
    value: {
      json: profileJson,
      url: 'ipfs://QmQTqheBLZFnQUxu5RDs8tA9JtkxfZqMBcmGd9sukXxwRm',
    },
  },
]);
/**
{
  keys: ['0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5'],
  values:
['0x6f357c6a2404a2866f05e53e141eb61382a045e53c2fc54831daca9d9e1e039a11f739e1696670733a2f2f5
16d5154716865424c5a466e5155787535524473387441394a746b78665a714d42636d47643973756b58787
7526d'],
}
*/

// You can also use the hashed key (with or without 0x prefix)
myErc725.encodeData([
  {
    keyName:
      '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
    value: {
      json: profileJson,
      url: 'ipfs://QmQTqheBLZFnQUxu5RDs8tA9JtkxfZqMBcmGd9sukXxwRm',
    },
  },
]);
/**
{
  keys: ['0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5'],
  values:
['0x6f357c6a2404a2866f05e53e141eb61382a045e53c2fc54831daca9d9e1e039a11f739e1696670733a2f2f5
16d5154716865424c5a466e5155787535524473387441394a746b78665a714d42636d47643973756b58787
7526d'],
}
*/

myErc725.encodeData([
```

```
  {
    keyName: '5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
    value: {
      json: profileJson,
      url: 'ipfs://QmQTqheBLZFnQUxu5RDs8tA9JtkxfZqMBcmGd9sukXxwRm',
    },
  },
]);
/**
{
  keys: ['0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5'],
  values:
['0x6f357c6a2404a2866f05e53e141eb61382a045e53c2fc54831daca9d9e1e039a11f739e1696670733a2f2f5
16d5154716865424c5a466e5155787535524473387441394a746b78665a714d42636d47643973756b58787
7526d'],
}
*/
```

```
myErc725.encodeData([
  {
    keyName: 'LSP1UniversalReceiverDelegate',
    value: '0x1183790f29BE3cDfD0A102862fEA1a4a30b3AdAb',
  },
]);
/**
{
  keys: ['0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47'],
  values: ['0x1183790f29be3cdfd0a102862fea1a4a30b3adab'],
}
*/
```

</details>

<details>
<summary>Encode a <code>JSONURL</code> with hash function, hash and uploaded URL</summary>

```
myErc725.encodeData([
  {
    keyName: 'LSP3Profile',
    value: {
      verification: {
        method: 'keccak256(utf8)',
        data: '0x820464ddfac1bec070cc14a8daf04129871d458f2ca94368aae8391311af6361',
      },
      url: 'ipfs://QmYr1VJLwerg6pEoscdhVGugo39pa6rycEZLjtRPDfW84UAx',
    },
  },
]);
/**
{
  keys: ['0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5'],
  values:
['0x6f357c6a820464ddfac1bec070cc14a8daf04129871d458f2ca94368aae8391311af6361696670733a2f2f5
16d597231564a4c776572673670456f736364685647756 76f33397061367279634 55a4c6a74525044665738
34554178'],
}
*/
```

</details>

<details>
<summary>Encode dynamic keys</summary>

```
const schemas = [
  {
    name: 'DynamicKey:<address>',
    key: '0x0fb367364e1852abc5f20000<address>',
    keyType: 'Mapping',
    valueType: 'bytes',
    valueContent: 'Address',
  },
];

myErc725.encodeData(
  [
    {
      keyName: 'DynamicKey:<address>',
      dynamicKeyParts: ['0xbedbedbedbedbedbedbedbedbedbedbedbedb'],
      value: '0xcafecafecafecafecafecafecafecafecafecafe',
    },
  ],
  schemas,
);
/**
{
  keys: ['0x0fb367364e1852abc5f20000bedbedbedbedbedbedbedbedbedbedbedbedb'],
  values: ['0xcafecafecafecafecafecafecafecafecafecafe]
}
```

```
*/

const schemas = [
  {
    name: 'DynamicKey:<bytes4>:<string>',
    key: '0xForDynamicKeysThisFieldIsIrrelevantAndWillBeOverwriten',
    keyType: 'Mapping',
    valueType: 'bytes',
    valueContent: 'Address',
  },
];

myErc725.encodeData(
  [
    {
      keyName: 'DynamicKey:<bytes4>:<string>',
      dynamicKeyParts: ['0x11223344', 'Summer'],
      value: '0xcafecafecafecafecafecafecafecafecafecafe',
    },
  ],
  schemas,
);
/**
{
  keys: ['0x0fb367364e1852abc5f2000078c964cd805233eb39f2db152340079088809725'],
  values: ['0xcafecafecafecafecafecafecafecafecafecafe']
}
*/
```

</details>

<details>
<summary>Encode multiple keys at once</summary>

```
myErc725.encodeData([
  {
    keyName: 'LSP3Profile',
    value: {
      verification: {
        method: 'keccak256(utf8)',
        data: '0x820464ddfac1bec070cc14a8daf04129871d458f2ca94368aae8391311af6361',
      },
      url: 'ipfs://QmYr1VJLwerg6pEoscdhVGugo39pa6rycEZLjtRPDfW84UAx',
    },
  },
  {
    keyName: 'LSP12IssuedAssets[]',
    value: [
      '0xD94353D9B005B3c0A9Da169b768a31C57844e490',
      '0xDaea594E385Fc724449E3118B2Db7E86dFBa1826',
    ],
  },
  {
    keyName: 'LSP1UniversalReceiverDelegate',
    value: '0x1183790f29BE3cDfD0A102862fEA1a4a30b3AdAb',
  },
]);
/**
{
  keys: [
    '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
    '0x7c8c3416d6cda87cd42c71ea1843df28ac4850354f988d55ee2eaa47b6dc05cd',
    '0x7c8c3416d6cda87cd42c71ea1843df2800000000000000000000000000000000',
    '0x7c8c3416d6cda87cd42c71ea1843df2800000000000000000000000000000001',
    '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
  ],
  values: [

'0x6f357c6a820464ddfac1bec070cc14a8daf04129871d458f2ca94368aae8391311af6361696670733a2f2f516d597231564a4c776572673670456f736364685647756 76f33397061367279634 55a4c6a7452504466573834554178',
    '0x00000000000000000000000000000002',
    '0xd94353d9b005b3c0a9da169b768a31c57844e490',
    '0xdaea594e385fc724449e3118b2db7e86dfba1826',
    '0x1183790f29be3cdfd0a102862fea1a4a30b3adab',
  ],
}
*/
```

</details>

---

encodeKeyName

```
ERC725.encodeKeyName(keyName [, dynamicKeyParts]);
```

Hashes a key name for use on an ERC725Y contract (https://github.com/ethereum/EIPs/blob/master/EIPS/eip-725.md#erc725y) according to the LSP2 ERC725Y JSON Schema Standard (https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-2-ERC725YJSONSchema.md).

:::info

encodeKeyName is available as either a static or non-static method so can be called without instantiating an ERC725 object.

:::

Parameters

1. keyName - String

The key name you want to encode, for instance: LSP3Profile.

2. dynamicKeyParts - String or array of Strings (optional)

The variables used to encode the key name, if the key name is a dynamic (i.e.: MyKey:<address>...)

Returns

Name	Type	Description
encodedKeyName	string	The keccak256 hash of the provided key name.

The hash must be retrievable from the ERC725Y contract via the getData function.

Example

```
ERC725.encodeKeyName('LSP3Profile');
// '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5'

ERC725.encodeKeyName('SupportedStandards:LSP3Profile');
// '0xeafec4d89fa9619884b600005ef83ad9559033e6e941db7d7c495acdce616347'

ERC725.encodeKeyName(
  'AddressPermissions:Permissions:cafecafecafecafecafecafecafecafecafecafe',
);
// '0x4b80742de2bf82acb3630000cafecafecafecafecafecafecafecafecafecafe'

ERC725.encodeKeyName('MyKeyName:<bool>', 'true');
// '0x35e6950bc8d21a1699e5000000000000000000000000000000000000000000001'

ERC725.encodeKeyName('MyKeyName:<bytes2>:<uint32>', ['ffff', '4081242941']);
// 0x35e6950bc8d20000ffff0000000000000000000000000000000000000f342d33d

ERC725.encodeKeyName('MyKeyName:<uint32>', ['4081242941']);
// 0x35e6950bc8d21a1699e5000000000000000000000000000000000000f342d33d

// This method is also available on the instance:
myErc725.encodeKeyName('LSP3Profile');
```

decodeMappingKey

```
ERC725.decodeMappingKey(keyNameOrSchema, keyHash);
```

Decode the values of the dynamic parts of a hashed key used on an ERC725Y contract (https://github.com/ethereum/EIPs/blob/master/EIPS/eip-725.md#erc725y) according to the LSP2 ERC725Y JSON Schema Standard (https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-2-ERC725YJSONSchema.md).

:::info

decodeMappingKey is available as either a static or non-static method so can be called without instantiating an ERC725 object.

:::

Parameters

1. keyHash - String

The bytes32 key hash that needs to be decoded

2. keyNameOrSchema - String or ERC725JSONSchema

The key name or erc725y json schema associated to the key hash to decode, for instance: MySuperKey:<address>.

Returns

Name	Type	Description
dynamicKeyParts	{type: string, value: string OR number OR boolean}[]	The dynamic key parts decoded from the key hash.

Example

```
ERC725.decodeMappingKey(
  '0x35e6950bc8d21a1699e50000cafecafecafecafecafecafecafecafecafecafe',
  'MyKeyName:<address>',
);
// Decoding will checksum addresses
// [{
//   type: 'address',
//   value: '0xCAfEcAfeCAfECaFeCaFecaFecaFECafECafeCaFe'
// }]

ERC725.decodeMappingKey(
  '0x35e6950bc8d21a1699e50000000000000000000000000000000000000000f342d33d',
  'MyKeyName:<uint32>',
);
// [{ type: 'uint32', value: 4081242941 }]

ERC725.decodeMappingKey(
  '0x35e6950bc8d21a1699e50000000000000000000000000000000000000000abcd1234',
  'MyKeyName:<bytes4>',
);
// [{ type: 'bytes4', value: 'abcd1234' }]

ERC725.decodeMappingKey(
  '0x35e6950bc8d21a1699e50000aaaabbbbccccddddeeeeffff1111222233334444',
  'MyKeyName:<bytes32>',
);
// [{
//   type: 'bytes32',
//   value: 'aaaabbbbccccddddeeeeffff1111222233334444'
// }]

ERC725.decodeMappingKey(
  '0x35e6950bc8d21a1699e50000000000000000000000000000000000000000000001',
  'MyKeyName:<bool>',
);
// [{ type: 'bool', value: true }]

ERC725.decodeMappingKey(
  '0x35e6950bc8d20000ffff0000000000000000000000000000000000000000f342d33d',
  'MyKeyName:<bytes2>:<uint32>',
);
// [
//   { type: 'bytes2', value: 'ffff' },
//   { type: 'uint32', value: 4081242941 }
// ]

// This method is also available on the instance:
myErc725.decodeMappingKey(
  '0x35e6950bc8d20000ffff0000000000000000000000000000000000000000f342d33d',
  'MyKeyName:<bytes2>:<uint32>',
);
```

encodePermissions

```
ERC725.encodePermissions(permissions);
```

Encodes permissions into a hexadecimal string as defined by the LSP6 KeyManager Standard (https://docs.lukso.tech/standards/universal-profile/lsp6-key-manager).

:::info

encodePermissions is available as either a static or non-static method so can be called without instantiating an ERC725 object.

:::

Parameters

1. permissions - Object

An object with LSP6 KeyManager Permissions (../../../../../standards/universal-profile/lsp6-key-manager#permissions) as keys and a boolean as value. Any ommited permissions will default to false.

Returns

Type	Description
string	The permissions encoded as a hexadecimal string defined by the LSP6 KeyManager Standard (https://docs.lukso.tech/standards/universal-profile/lsp6-key-manager)

Example

```
ERC725.encodePermissions({
  CHANGEOWNER: false,
  ADDCONTROLLER: false,
  EDITPERMISSIONS: false,
  ADDEXTENSIONS: false,
  CHANGEEXTENSIONS: true,
  ADDUNIVERSALRECEIVERDELEGATE: false,
  CHANGEUNIVERSALRECEIVERDELEGATE: false,
  REENTRANCY: false,
  SUPER_TRANSFERVALUE: true,
  TRANSFERVALUE: true,
  SUPER_CALL: false,
  CALL: true,
  SUPER_STATICCALL: false,
  STATICCALL: false,
  SUPER_DELEGATECALL: false,
  DELEGATECALL: false,
  DEPLOY: false,
  SUPER_SETDATA: false,
  SETDATA: false,
  ENCRYPT: false,
  DECRYPT: false,
  SIGN: false,
  EXECUTE_RELAY_CALL: false
}),
// '0x0000000000000000000000000000000000000000000000000000000000000110'

// Any ommited Permissions will default to false
ERC725.encodePermissions({
  ADDCONTROLLER: true,
  ADDEXTENSIONS: true,
}),
// '0x000000000000000000000000000000000000000000000000000000000000000a'
ERC725.encodePermissions({
  EDITPERMISSIONS: true,
  CHANGEEXTENSIONS: true,
  CHANGEUNIVERSALRECEIVERDELEGATE: true,
  SETDATA: true,
}),
// '0x0000000000000000000000000000000000000000000000000000000000040054'


// This method is also available on the instance:
myErc725.encodePermissions({
  EDITPERMISSIONS: true,
  SETDATA: true,
}),
```

fetchData

```
myErc725.fetchData([keys]);
```

The fetchData function fetches smart contract data and can additionally return JSONURL (https://github.com/lukso-network/LIPs/blob/master/LSPs/LSP-2-ERC725YJSONSchema.md#jsonurl) or ASSETURL (https://github.com/lukso-network/LIPs/blob/master/LSPs/LSP-2-ERC725YJSONSchema.md#asseturl) data from IPFS, HTTP, or HTTPS endpoints.

:::info

To ensure data authenticity fetchData compares the hash of the fetched JSON with the hash stored on the blockchain.

:::

:::info

If you get the ReferenceError: fetch is not defined error, you need to install and import isomorphic-fetch (https://www.npmjs.com/package/isomorphic-fetch).

:::

Parameters

1. keys - String, Object or array of Strings or/and Objects (optional)

The name(s) (or the encoded name(s) as schema key) of the element(s) in the smart contract's schema. If no keys are set, it will fetch all the non dynamic schema keys given at instantiation. For dynamic keys, you can use the object below:

Name	Type	Description
keyName	string	The dynamic key name, such as MyKey:<address>
dynamicKeyParts	string or string[ ]	The dynamic parts of the keyName that will be used for encoding the key.

Returns

Name	Type	Description
data	Promise<Array> or Promise<Object>	An array with same objects as for decodeData() (./ERC725#decodedata) function but with the value being remplaced by the actual file for JSONURL and ASSETURL valueContent. If there is a hash mismatch, the value will be null.

:::info

- If the input is an array, the values will be returned in an array.
- If the input is a single key, the output will be the object directly.

:::

All-Keys Example

```
await myErc725.fetchData();
/**
[
  {
    name: 'SupportedStandards:LSP3Profile',
    key: '0xeafec4d89fa9619884b600005ef83ad9559033e6e941db7d7c495acdce616347',
    value: '0x5ef83ad9'
  },
  {
    name: 'LSP3Profile',
    key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
    value: { LSP3Profile: [Object] }
  },
  {
    name: 'LSP1UniversalReceiverDelegate',
    key: '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
    value: '0xE2D6038acD92200790Df695Ebd13856CdF2a6942'
  },
  {
    name: 'LSP12IssuedAssets[]',
    key: '0x7c8c3416d6cda87cd42c71ea1843df28ac4850354f988d55ee2eaa47b6dc05cd',
    value: [
      '0xc444009d38d3046bb0cF81Fa2Cd295ce46A67C78',
      '0x4fEbC3491230571F6e1829E46602e3b110215A2E',
      '0xB92a8DdA288638491AEE5C2a003D4CAbfa47aE3F',
      '0x1e52e7F1707dcda57dD33F003B2311652A465acA',
      '0x0BDA71aA980D37Ea56E8a3784E4c309101DAf3E4',
      '0xfDB4D9C299438B9839e9d04E34B9609C5b56600D',
      '0x081D3F0bff8ae2339cb65113822eEc1510704d5c',
      '0x55C98c6944B7497FaAf4db0386a1aD1E6efF526E',
      '0x90D1a1D68fa23AEEE991220703f1a1C3782e0b35',
      '0xdB5AB19792d9fB61c1Dff57810Fb7C6f839Af8ED'
    ]
  }
]
*/
```

Single-Key Example

```
await myErc725.fetchData('LSP3Profile');
/**
{
  key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
  name: 'LSP3Profile',
  value: { LSP3Profile: { name: 'Test', description: 'Cool' } }
}
*/
await myErc725.fetchData(['LSP1UniversalReceiverDelegate']);
/**
[
  {
    name: 'LSP1UniversalReceiverDelegate',
    key: '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
    value: '0xE2D6038acD92200790Df695Ebd13856CdF2a6942'
  }
]
*/
```

Multi-Keys / Dynamic-Keys Example

```
await myErc725.fetchData(['LSP3Profile', 'LSP1UniversalReceiverDelegate']);
/**
[
  {
    name: 'LSP3Profile',
    key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
    value: { LSP3Profile: [Object] }
  },
  {
    name: 'LSP1UniversalReceiverDelegate',
    key: '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
    value: '0xE2D6038acD92200790Df695Ebd13856CdF2a6942'
  }
]
*/

await myErc725.fetchData([
  'LSP1UniversalReceiverDelegate',
  {
    keyName: 'LSP12IssuedAssetsMap:<address>',
    dynamicKeyParts: '0xcafecafecafecafecafecafecafecafecafecafe',
  },
]);
/**
[
  {
    name: 'LSP1UniversalReceiverDelegate',
    key: '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
    value: '0xE2D6038acD92200790Df695Ebd13856CdF2a6942'
  },
  {
    name: 'LSP12IssuedAssetsMap:cafecafecafecafecafecafecafecafecafecafe',
    key: '0x74ac2555c10b9349e78f0000cafecafecafecafecafecafecafecafecafecafe',
    value: null
  }
]
*/
```

---

getData

```
myErc725.getData([keys]);
```

Gets decoded data for one, many, or all of the specified ERC725 smart contract's keys.
When omitting the keys parameter, it will give back every key (as per ERC725JSONSchema definition).

:::caution

- Data returned by this function does not contain external data of [JSONURL (https://github.com/lukso-network/LIPs/blob/master/LSPs/LSP-2-ERC725YJSONSchema.md#jsonurl)](https://github.com/lukso-network/LIPs/blob/master/LSPs/LSP-2-ERC725YJSONSchema.md#jsonurl)
  or [ASSETURL (https://github.com/lukso-network/LIPs/blob/master/LSPs/LSP-2-](https://github.com/lukso-network/LIPs/blob/master/LSPs/LSP-2-)

ERC725YJSONSchema.md#asseturl) schema elements.
- If you would like to receive everything in one go, you can use fetchData (ERC725.md#fetchdata).

:::

Parameters

1. keys - String, Object or array of Strings or/and Objects (optional)

The name(s) (or the encoded name(s) as schema key) of the element(s) in the smart contract's schema. If no keys are set, it will fetch all the non dynamic schema keys given at instantiation. For dynamic keys, you can use the object below:

Name	Type	Description
keyName	string	The dynamic key name, such as MyKey:<address>
dynamicKeyParts	string or string[ ]	The dynamic parts of the keyName that will be used for encoding the key.

Returns

Name	Type	Description
data	Promise<Array> or Promise<Object>	An array with same objects as for decodeData() (./ERC725#decodedata) function.

:::info

- If the input is an array, the values will be returned in an array.
- If the input is a single key, the output will be the object directly.

:::

All-Keys Example

```
await myErc725.getData();
/**
[
  {
    name: 'SupportedStandards:LSP3Profile',
    key: '0xeafec4d89fa9619884b600005ef83ad9559033e6e941db7d7c495acdce616347',
    value: '0x5ef83ad9',
  },
  {
    name: 'LSP1UniversalReceiverDelegate',
    key: '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
    value: '0x50A02EF693fF6961A7F9178d1e53CC8BbE1DaD68',
  },
  {
    name: 'LSP12IssuedAssets[]',
    key: '0x7c8c3416d6cda87cd42c71ea1843df28ac4850354f988d55ee2eaa47b6dc05cd',
    value: [
      '0xc444009d38d3046bb0cF81Fa2Cd295ce46A67C78',
      '0x4fEbC3491230571F6e1829E46602e3b110215A2E',
      '0xB92a8DdA288638491AEE5C2a003D4CAbfa47aE3F',
      '0x1e52e7F1707dcda57dD33F003B2311652A465acA',
      '0x0BDA71aA980D37Ea56E8a3784E4c309101DAf3E4',
      '0xfDB4D9C299438B9839e9d04E34B9609C5b56600D',
      '0x081D3F0bff8ae2339cb65113822eEc1510704d5c',
      '0x55C98c6944B7497FaAf4db0386a1aD1E6efF526E',
      '0x90D1a1D68fa23AEEE991220703f1a1C3782e0b35',
      '0xdB5AB19792d9fB61c1Dff57810Fb7C6f839Af8ED'
    ],
  },
  {
    name: 'LSP3Profile',
    key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
    value: {
      verification: {
        method: 'keccak256(utf8)',
        data: '0x70546a2accab18748420b63c63b5af4cf710848ae83afc0c51dd8ad17fb5e8b3',
      },
      url: 'ipfs://QmecrGejUQVXpW4zS948pNvcnQrJ1KiAoM6bdfrVcWZsn5',
    },
  },
]
*/
```

Single-Key Example

```
await myErc725.getData('LSP3Profile');
/**
{
  name: 'LSP3Profile',
  key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
  value: {
    verification: {
      method: 'keccak256(utf8)',
      data: '0xd96ff7776660095f661d16010c4349aa7478a9129ce0670f771596a6ff2d864a',
    },
    url: 'ipfs://QmbTmcbp8ZW23vkQrqkasMFqNg2z1iP4e3BCUMz9PKDsSV'
  },
}
*/

await myErc725.getData(['LSP3Profile']);
/**
[
  {
    name: 'LSP3Profile',
    key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
    value: {
      verification: {
        method: 'keccak256(utf8)',
        data: '0xd96ff7776660095f661d16010c4349aa7478a9129ce0670f771596a6ff2d864a',
      },
      url: 'ipfs://QmbTmcbp8ZW23vkQrqkasMFqNg2z1iP4e3BCUMz9PKDsSV'
    },
  }
]
*/

await myErc725.getData('LSP1UniversalReceiverDelegate');
/**
{
  name: 'LSP1UniversalReceiverDelegate',
  key: '0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
  value: '0x50A02EF693fF6961A7F9178d1e53CC8BbE1DaD68',
}
*/
```

Multi-Key Example

```
await myErc725.getData(['LSP3Profile', 'LSP1UniversalReceiverDelegate']);
/**
[
  {
    name: 'LSP3Profile',
    key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
    value: {
      verification: {
        method: 'keccak256(utf8)',
        data: '0xeeafeebeb416923dfb0dcf4c66b045c72742121ce2a06f93ae044ee0efb70777',
      },
      url: 'ipfs://QmZnG5Z5B5Dq8iFFtsL5i7AnrgH16P4DJ8UhY7j8RzX51p'
    }
  },
  {
    name: 'LSP1UniversalReceiverDelegate',
    key: '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
    value: '0xE2D6038acD92200790Df695Ebd13856CdF2a6942'
  }
]
*/
```

Dynamic-Key Example

```
await myErc725.getData({
  keyName: 'LSP12IssuedAssetsMap:<address>',
  dynamicKeyParts: '0xcafecafecafecafecafecafecafecafecafecafe',
});
/**
{
  name: 'LSP12IssuedAssetsMap:cafecafecafecafecafecafecafecafecafecafe',
  key: '0x74ac2555c10b9349e78f0000cafecafecafecafecafecafecafecafecafecafe',
  value: '0x6b175474e89094c44da98b954eedeac495271d0f',
}
*/

await myErc725.getData([
  {
    keyName: 'LSP12IssuedAssetsMap:<address>',
    dynamicKeyParts: '0xcafecafecafecafecafecafecafecafecafecafe',
  },
]);
/**
[
  {
    name: 'LSP12IssuedAssetsMap:cafecafecafecafecafecafecafecafecafecafe',
    key: '0x74ac2555c10b9349e78f0000cafecafecafecafecafecafecafecafecafecafe',
    value: '0x6b175474e89094c44da98b954eedeac495271d0f',
  }
]
*/
```

```
await myErc725.getData([
  'LSP3Profile',
  {
    keyName: 'LSP12IssuedAssetsMap:<address>',
    dynamicKeyParts: '0xcafecafecafecafecafecafecafecafecafecafe',
  },
]);
/**
[
  {
    name: 'LSP3Profile',
    key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
    value: {
      verification: {
        method: 'keccak256(utf8)',
        data: '0xeeafeebeb416923dfb0dcf4c66b045c72742121ce2a06f93ae044ee0efb70777',
      },
      url: 'ipfs://QmZnG5Z5B5Dq8iFFtsL5i7AnrgH16P4DJ8UhY7j8RzX51p'
    }
  },
  {
    name: 'LSP12IssuedAssetsMap:cafecafecafecafecafecafecafecafecafecafe',
    key: '0x74ac2555c10b9349e78f0000cafecafecafecafecafecafecafecafecafecafe',
    value: null
  }
]
*/
```

---

getOwner

```
myErc725.getOwner([address]);
```

Returns the contract owner and is not directly related to ERC725 specifications.

Parameters

1. address - String (optional)

The contract address you wish to find the owner of. If no address is set, will return the owner of the contract used to initialise the ERC725 class.

Returns

Name	Type	Description
Promise	string	The contract or EOA address of the owner.

:::info

The address of the contract owner as stored in the contract.

:::

Example

```
// If no address is set, it will return the owner of the contract used to initialise the ERC725() class.
await myErc725.getOwner();
// '0x94933413384997F9402cc07a650e8A34d60F437A'

// You can also get the owner of a specific contract by setting the address parameter
await myErc725.getOwner('0x3000783905Cc7170cCCe49a4112Deda952DDBe24');
// '0x7f1b797b2Ba023Da2482654b50724e92EB5a7091'
```

---

getSchema

```
myErc725.getSchema(keys [, providedSchemas]);
```

Parses a hashed key or a list of hashed keys and will attempt to return its corresponding LSP2 ERC725YJSONSchema (https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-2-ERC725YJSONSchema.md) object. Additionally, it will look for a corresponding key within the schemas:

- in the schemas (https://github.com/ERC725Alliance/myErc725.js/tree/main/schemas) folder (which includes all LSPs (https://github.com/lukso-network/LIPs/tree/main/LSPs)),
- that were provided at ERC725 initialisation, and
- that were provided in the function call (providedSchemas).

Parameters

1. keys - String or array of Strings

The key(s) you are trying to get the schema for.

2. providedSchemas - Object (optional)

An array of extra LSP-2 ERC725YJSONSchema (https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-2-ERC725YJSONSchema.md) objects that can be used to find the schema.

Returns

Name	Type	Description
result	ERC725JSONSchema	If the parameter keys is a string and the schema was found.
result	Record string	If the parameter keys is a string[ ] and the schema was found.
result	null	If the schema was not found.

Example using a predefined LSP3 schema

```
myErc725.getSchema(
  '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
);
/**
{
  name: 'LSP3Profile',
  key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
  keyType: 'Singleton',
  valueContent: 'JSONURL',
  valueType: 'bytes'
}
*/
myErc725.getSchema([
  '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
  '0x3a47ab5bd3a594c3a8995f8fa58d0876000000000000000000000000000000000001',
]);
/**
{
  '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5': {
    name: 'LSP3Profile',
    key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
    keyType: 'Singleton',
    valueContent: 'JSONURL',
    valueType: 'bytes'
  },
  '0x3a47ab5bd3a594c3a8995f8fa58d0876000000000000000000000000000000000001': {
    name: 'LSP12IssuedAssets[1]',
    key: '0x7c8c3416d6cda87cd42c71ea1843df28ac4850354f988d55ee2eaa47b6dc05cd',
    keyType: 'Singleton',
    valueContent: 'Address',
    valueType: 'address'
  }
}
*/
```

Example using a custom schema

```
myErc725.getSchema(
  '0x777f55baf2e0c9f73d3bb456dfb8dbf6e609bf557969e3184c17ff925b3c402c',
  [
    {
      name: 'ParameterSchema',
      key: '0x777f55baf2e0c9f73d3bb456dfb8dbf6e609bf557969e3184c17ff925b3c402c',
      keyType: 'Singleton',
      valueContent: 'JSONURL',
      valueType: 'bytes',
    },
  ],
);
/**
{
  name: 'ParameterSchema',
  key: '0x777f55baf2e0c9f73d3bb456dfb8dbf6e609bf557969e3184c17ff925b3c402c',
  keyType: 'Singleton',
  valueContent: 'JSONURL',
  valueType: 'bytes',
}
*/
```

---

isValidSignature

```
myErc725.isValidSignature(messageOrHash, signature);
```

Checks if a signature was signed by the owner of the ERC725 Account contract, according to [EIP-1271 (https://eips.ethereum.org/EIPS/eip-1271)](). If the owner is a contract itself, it will delegate the isValidsignature() call to the owner contract if it supports [EIP-1271 (https://eips.ethereum.org/EIPS/eip-1271)](). Otherwise, it will fail.

Parameters

1. messageOrHash - String

Value of a message or hash that needs to be verified.

2. signature - String

The raw RLP encoded signature.

:::info

- The hash must be 66 chars long with the 0x prefix, otherwise it will be interpreded as message.
- The message will be: enveloped as "\x19Ethereum Signed Message:\n" + message.length + message and hashed using keccak256 function.
  The signature can be generated with [web3.eth.accounts.sign() (https://web3js.readthedocs.io/en/v1.2.11/web3-eth-accounts.html#sign)]().

:::

Returns

Name	Type	Description
Promise	boolean	true if signature is valid, false if signature is invalid.

:::info

- A valid signature means that the smart contract response IS the MAGICVALUE: 0x1626ba7e.
- If this function is called on a contract which does not support [EIP-1271 (https://eips.ethereum.org/EIPS/eip-1271)](), it will throw an error.

:::

Examples

```
await myErc725.isValidSignature(
  'hello',

'0xb91467e570a6466aa9e9876cbcd013baba02900b8979d43fe208a4a4f339f5fd6007e74cd82e037b800186422fc2da167c747ef045e5d18a5f5d4300f8e1a0291c',
);
// true
```

```
await myErc725.isValidSignature(
  '0x1da44b586eb0729ff70a73c326926f6ed5a25f5b056e7f47fbc6e58d86871655',

'0xcafecafeb915466aa9e9876cbcd013baba02900b8979d43fe208a4a4f339f5fd6007e74cd82e037b800186422fc2da167c747ef045e5d18a5f5d4300f8e1a0291c',
);
// false
```

supportsInterface

```
myERC725.supportsInterface(interfaceIdOrName);
```

```
ERC725.supportsInterface(interfaceIdOrName, options);
```

You can use this function if you need to check if the ERC725 object or a smart contract supports a specific interface (by ID or name). When you use the function on your instantiated ERC725 class, it will use the contract address and provider provided at instantiation. On non instantiated class, you need to specify them in the options parameter.

:::caution
The interfaceId is not the most secure way to check for a standard, as they could be set manually.
:::

Parameters

1. interfaceIdOrName - String

Either a string of the hexadecimal interfaceID as defined by [ERC165 (https://eips.ethereum.org/EIPS/eip-165)]() or one of the predefined interface names:

interfaceName	Standard
ERC1271	[EIP-1271: Standard Signature Validation Method for Contracts]()

	(https://eips.ethereum.org/EIPS/eip-1271)
ERC725X	EIP-725: General execution standard (https://eips.ethereum.org/EIPS/eip-725)
ERC725Y	EIP-725: General key-value store (https://eips.ethereum.org/EIPS/eip-725)
LSP0ERC725Account	LSP-0: ERC725 Account (https://docs.lukso.tech/standards/universal-profile/lsp0-erc725account)
LSP1UniversalReceiver	LSP-1: Universal Receiver (https://docs.lukso.tech/standards/generic-standards/lsp1-universal-receiver)
LSP1UniversalReceiverDelegate	LSP-1: Universal Receiver Delegate (https://docs.lukso.tech/standards/universal-profile/lsp1-universal-receiver-delegate)
LSP6KeyManager	LSP-6: Key Manager (https://docs.lukso.tech/standards/universal-profile/lsp6-key-manager)
LSP7DigitalAsset	LSP-7: Digital Asset (https://docs.lukso.tech/standards/nft-2.0/LSP7-Digital-Asset)
LSP8IdentifiableDigitalAsset	LSP-8: Identifiable Digital Asset (https://docs.lukso.tech/standards/nft-2.0/LSP8-Identifiable-Digital-Asset)
LSP9Vault	LSP-9: Vault (https://docs.lukso.tech/standards/universal-profile/lsp9-vault)

:::info

The interfaceName will only check for the latest version of the standard's interfaceID, which can be found in src/constants/interfaces. For LSPs, the interfaceIDs are taken from the latest release of the @lukso/lsp-smart-contracts (https://github.com/lukso-network/lsp-smart-contracts) library.

:::info

2. options - Object (optional)

On non instantiated class, you should provide an options object.

Name	Type	Description
address	string	Address of the smart contract to check against a certain interface.
rpcUrl	string	RPC URL to connect to the network the smart contract is deployed to.
gas	number	Optional: gas parameter to use. Default: 1_000_000.

Returns

Type	Description
Promise<boolean>	Returns true if the interface was found, otherwise false.

Examples

```
myErc725.supportsInterface('0xfd4d5c50');
// true

ERC725.supportsInterface('0xfd4d5c50', {
  address: '0xe408BDDbBAB1985006A2c481700DD473F932e5cB',
  rpcUrl: 'https://rpc.testnet.lukso.network',
});
// false
```

```
myErc725.supportsInterface('LSP0ERC725Account');
// false

ERC725.supportsInterface('LSP0ERC725Account', {
  address: '0x0Dc07C77985fE31996Ed612F568eb441afe5768D',
  rpcUrl: 'https://rpc.testnet.lukso.network',
  gas: 20_000_000,
});
// true
```

</file>

<file>

path: /docs/getting-started.md

url: https://github.com/ERC725Alliance/erc725.js/blob/main/docs/getting-started.md

---

sidebar_position: 1

Getting Started

:::caution
This package is currently in the early stages of development. Please use it for testing or experimentation purposes only.
:::

The @erc725/erc725.js package allows you to interact with the ERC-725 schemas easily.

- GitHub repo: https://github.com/ERC725Alliance/erc725.js
- NPM: https://www.npmjs.com/package/@erc725/erc725.js

Installation

```
npm install @erc725/erc725.js
```

:::info

If you install it on the backend side, you may need to also install [isomorphic-fetch (https://www.npmjs.com/package/isomorphic-fetch)](https://www.npmjs.com/package/isomorphic-fetch).

:::

Instantiation

```
import { ERC725 } from '@erc725/erc725.js';
import Web3 from 'web3';

// Part of LSP3-UniversalProfile Schema
// https://github.com/lukso-network/LIPs/blob/master/LSPs/LSP-3-UniversalProfile.md
const schemas = [
  {
    name: 'SupportedStandards:LSP3Profile',
    key: '0xeafec4d89fa9619884b600005ef83ad9559033e6e941db7d7c495acdce616347',
    keyType: 'Mapping',
    valueType: 'bytes',
    valueContent: '0x5ef83ad9',
  },
  {
    name: 'LSP3Profile',
    key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
    keyType: 'Singleton',
    valueType: 'bytes',
    valueContent: 'JSONURL',
  },
  {
    name: 'LSP1UniversalReceiverDelegate',
    key: '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
    keyType: 'Singleton',
    valueType: 'address',
    valueContent: 'Address',
  },
];

const address = '0x0Dc07C77985fE31996Ed612F568eb441afe5768D';
const RPC_URL = 'https://rpc.testnet.lukso.network';
const config = {
  ipfsGateway: 'https://YOUR-IPFS-GATEWAY/ipfs/',
  gas: 20_000_000, // optional, default is 1_000_000
};

const erc725 = new ERC725(schemas, address, RPC_URL, config);
```

Usage

```
await erc725.getOwner();
// > '0x28D25E70819140daF65b724158D00c373D1a18ee'

await erc725.getData('SupportedStandards:LSP3Profile');
/**
{
  'SupportedStandards:LSP3Profile': '0x5ef83ad9'
}
*/

await erc725.getData(['LSP3Profile', 'SupportedStandards:LSP3Profile']);
/**
{
  LSP3Profile: {
    url: 'ipfs://QmXybv2LdJWscy1C6yRKUjvnaj6aqKktZX4g4xmz2nyYj2',
    verification: {
      data: '0xb4f9d72e83bbe7e250ed9ec80332c493b7b3d73e0d72f7b2c7ab01c39216eb1a',
      method: 'keccak256(utf8)'
    }
  },
  'SupportedStandards:LSP3Profile': '0x5ef83ad9'
}
*/

await erc725.fetchData('LSP3Profile'); // downloads and verifies the linked JSON
/**
{
  LSP3Profile: {
    LSP3Profile: {
        name: 'frozeman',
        description: 'The inventor of ERC725 and ERC20...',
        links: [
            { title: 'Twitter', url: 'https://twitter.com/feindura' },
            { title: 'lukso.network', url: 'https://lukso.network' }
        ],
        ...
    }
  }
}
*/
```

:::tip Try it out
You can run the code snippet within your browser using the corresponding StackBlitz example (https://stackblitz.com/edit/erc725js-instantiation?devtoolsheight=66&file=index.js).

:::note
Whenever you can you should import ERC725 via the named export. However currently we are also providing a default export.

```
import ERC725 from 'erc725.js';
```

:::

After the instance has been created, it is still possible to change settings through the options property.

```
myERC725.options.schema = '<schema>' // change schema
myERC725.options.address '<address>' // change address
myERC725.options.ipfsGateway = '<url>' // used for fetchData(), default: 'https://cloudflare-ipfs.com/ipfs/'
myERC725.options.gas = 20_000_000 // change gas setting

// NOTE: ERC725.provider can not be changed
```

</file>

<file>

path: /docs/providers.md

url: https://github.com/ERC725Alliance/erc725.js/blob/main/docs/providers.md

---

sidebar_position: 3

Providers

The provider by which @erc725/erc725.js will request blockchain data is set on the instantiation of the class through the configuration object.

The following provider types are supported:

RPC URL

An RPC URL can be passed when instantiating the ERC725 class.

```
import ERC725 from '@erc725/erc725.js';

const RPC_URL = 'https://rpc.testnet.lukso.network';

const erc725 = new ERC725([], '0x...', RPC_URL);
```

Ethereum (injected provider from extension)

```
import ERC725 from '@erc725/erc725.js';

const ethereumProvider = window.ethereum;

const erc725 = new ERC725([], '0x...', ethereumProvider);
```

Web3 (deprecated)

The following code snippet will use the web3 provider available at web3.providers from the corresponding web3 library.

:::caution Warning

Web3.js providers are being deprecated. Please provide an RPC URL or injected Ethereum provider instead.

:::

The following code snippet will use the web3 provider available at web3.providers from the corresponding web3 library.

```
import Web3 from 'web3';
import ERC725 from '@erc725/erc725.js';

const web3provider = new Web3(
  new Web3.providers.HttpProvider('https://rpc.testnet.lukso.network'),
);

const erc725 = new ERC725([], '0x...', web3provider);
```

</file>

<file>

path: /docs/schemas.md

url: https://github.com/ERC725Alliance/erc725.js/blob/main/docs/schemas.md

---

sidebar_position: 2

Schemas

The @erc725/erc725.js library contains a range of standard LSP ERC725 JSON schemas (https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-2-ERC725YJSONSchema.md).

Schemas allow erc725.js to know how to decode and encode data written in an ERC725Y (https://eips.ethereum.org/EIPS/eip-725) smart contract.

A quick reference for keys used in schema definitions can be seen below

Official Documentation (https://github.com/lukso-network/LIPs/blob/master/LSPs/LSP-2-ERC725YJSONSchema.md).

- name: An arbitrary name
- key: The sha3 hash of the name
- keyType: One of the supported erc725 keyTypes
- valueType: The type of the content data in store for decoding
- valueContent: The described content type for parsing

Standard LSP Schemas

The most common schemas of LUKSO Standard Proposals (https://github.com/lukso-network/LIPs/tree/main/LSPs) are available under the schemas/ (https://github.com/ERC725Alliance/erc725.js/tree/develop/schemas) folder.

Current provided LSPs are:

```
LSP1UniversalReceiverDelegate.json
LSP3ProfileMetadata.json
LSP4DigitalAssetLegacy.json
LSP4DigitalAsset.json
LSP5ReceivedAssets.json
LSP6KeyManager.json
LSP8IdentifiableDigitalAsset.json
LSP9Vault.json
LSP10ReceivedVaults.json
LSP12IssuedAssets.json
LSP17ContractExtension.json
```

You can import them from:

```
import LSP3 from '@erc725/erc725.js/schemas/LSP3ProfileMetadata.json';
import LSP5 from '@erc725/erc725.js/schemas/LSP5ReceivedAssets.json';
// ...

// Later use them on instantiation
const myErc725Contract = new ERC725js(LSP3, address, web3.currentProvider);

// You can retrieve the current loaded schema via
myErc725Contract.options.schemas;
```

</file>

<file>

path: /examples/README.md

url: https://github.com/ERC725Alliance/erc725.js/blob/main/examples/README.md

Examples

fetchData

```
npm run fetchData
```

(https://stackblitz.com/edit/erc725js-fetch-data?devtoolsheight=66&file=index.js)

getData

```
npm run getData
```

(https://stackblitz.com/edit/erc725js-get-data?devtoolsheight=66&file=package.json)

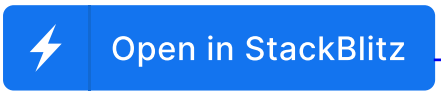
encodeData

```
npm run encodeData
```

(https://stackblitz.com/edit/erc725js-encode-data?devtoolsheight=66&file=index.js)

decodeData

```
npm run decodeData
```



(https://stackblitz.com/edit/erc725js-decode-data?devtoolsheight=66)

</file>

<file>

path: /examples/package.json

url: https://github.com/ERC725Alliance/erc725.js/blob/main/examples/package.json

```
{
  "name": "erc725.js-examples",
  "type": "module",
  "private": true,
  "scripts": {
    "fetchData": "node src/fetchData.js",
    "getData": "node src/getData.js",
    "encodeData": "node src/encodeData.js",
    "decodeData": "node src/decodeData.js"
  },
  "dependencies": {
    "@erc725/erc725.js": "^0.17.2",
    "web3": "^1.10.0"
  },
  "devDependencies": {
    "isomorphic-fetch": "^3.0.0"
  }
}
```

</file>

<file>

path: /examples/src/decodeData.js

url: https://github.com/ERC725Alliance/erc725.js/blob/main/examples/src/decodeData.js

```
// https://stackoverflow.com/questions/60059121/nodejs-es6-imports-cannot-find-
```

```
module#comment106219502_60059121
// eslint-disable-next-line import/extensions
import { getInstance } from './instantiation.js';

const myERC725 = getInstance();

const decodedDataOneKey = myERC725.decodeData([
  {
    keyName: 'LSP3Profile',
    value:

'0x6f357c6a820464ddfac1bec070cc14a8daf04129871d458f2ca94368aae8391311af6361697066733a2f2f516d597231564a4c776572673670456f736364685647756 76f3339706136727963455a4c6a745250446657383 4554178',
  },
]);
/**
[
  {
    name: 'LSP3Profile',
    key: '0x5ef83ad95590 33e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
    value: {
      verification: {
        method: 'keccak256(utf8)',
        data: '0x820464ddfac1bec070cc14a8daf04129871d458f2ca94368aae8391311af6361',
      },
      url: 'ipfs://QmYr1VJLwerg6pEoscdhVGugo39pa6rycEZLjtRPDfW84UAx'
    }
  }
]
*/

const decodedDataManyKeys = myERC725.decodeData([
  {
    keyName: 'LSP3Profile',
    value:

'0x6f357c6a820464ddfac1bec070cc14a8daf04129871d458f2ca94368aae8391311af6361697066733a2f2f516d597231564a4c776572673670456f736364685647756 76f3339706136727963455a4c6a745250446657383 4554178',
  },
  {
    keyName: 'LSP3IssuedAssets[]',
    value: [
      {
        key: '0x3a47ab5bd3a594c3a8995f8fa58d0876c96819ca4516bd76100c92462f2f9dc0',
        value:
          '0x0000000000000000000000000000000000000000000000000000000000000002',
      },
      {
        key: '0x3a47ab5bd3a594c3a8995f8fa58d0876000000000000000000000000000000000',
        value: '0xd94353d9b005b3c0a9da169b768a31c57844e490',
```

```
    },
    {
      key: '0x3a47ab5bd3a594c3a8995f8fa58d0876000000000000000000000000000000001',
      value: '0xdaea594e385fc724449e3118b2db7e86dfba1826',
    },
  ],
 },
]);
/**
[
 {
  name: 'LSP3Profile',
  key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
  value: {
   verification: {
    method: 'keccak256(utf8)',
    data: '0x820464ddfac1bec070cc14a8daf04129871d458f2ca94368aae8391311af6361',
   },
   url: 'ipfs://QmYr1VJLwerg6pEoscdhVGugo39pa6rycEZLjtRPDfW84UAx'
  }
 },
 {
  name: 'LSP3IssuedAssets[]',
  key: '0x3a47ab5bd3a594c3a8995f8fa58d0876c96819ca4516bd76100c92462f2f9dc0',
  value: [
   '0xD94353D9B005B3c0A9Da169b768a31C57844e490',
   '0xDaea594E385Fc724449E3118B2Db7E86dFBa1826'
  ]
 }
]
*/
console.log('/*-----------------------------------------*/');
console.log('/* decodeData - one key                    */');
console.log('/*-----------------------------------------*/');
console.log(decodedDataOneKey);
console.log('/*-----------------------------------------*/');
console.log('/* decodeData - many keys                  */');
console.log('/*-----------------------------------------*/');
console.log(decodedDataManyKeys);
```

</file>

<file>

path: /examples/src/encodeData.js

url: https://github.com/ERC725Alliance/erc725.js/blob/main/examples/src/encodeData.js

```
// https://stackoverflow.com/questions/60059121/nodejs-es6-imports-cannot-find-module#comment106219502_60059121
// eslint-disable-next-line import/extensions
```

```
import { getInstance, profileJson } from './instantiation.js';

const myERC725 = getInstance();

const encodedDataOneKey = myERC725.encodeData({
  keyName: 'LSP3Profile',
  value: {
    json: profileJson, // check instantiation.js to see the actual JSON
    url: 'ipfs://QmQTqheBLZFnQUxu5RDs8tA9JtkxfZqMBcmGd9sukXxwRm',
  },
});
/**
{
  keys: [
    '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5'
  ],
  values: [

'0x6f357c6a2404a2866f05e53e141eb61382a045e53c2fc548311daca9d9e1e039a11f739e1697066733a2f2f516d5154716865424c5a466e5155787535524473387441394a746b78665a714d42636d476439737563b5878775266d'
  ]
}
*/

const encodedDataOneKeyV2 = myERC725.encodeData({
  keyName: 'LSP3Profile',
  value: {
    verification: {
      method: 'keccak256(utf8)',
      data: '0x820464ddfac1bec070cc14a8daf04129871d458f2ca943688aae8391311af6361',
    },
    url: 'ipfs://QmYr1VJLwerg6pEoscdhVGugo39pa6rycEZLjtRPDfW84UAx',
  },
});
/**
{
  keys: [
    '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5'
  ],
  values: [

'0x6f357c6a820464ddfac1bec070cc14a8daf04129871d458f2ca943688aae8391311af6361697066733a2f2f516d597231564a4c776572673670456f736364685647756 76f3339706136727963455a4c6a745250446657383455417 8'
  ]
}
*/

const encodedDataManyKeys = myERC725.encodeData([
  {
    keyName: 'LSP3Profile',
```

```
    value: {
      verification: {
        method: 'keccak256(utf8)',
        data: '0x820464ddfac1bec070cc14a8daf04129871d458f2ca94368aae8391311af6361',
      },
      url: 'ipfs://QmYr1VJLwerg6pEoscdhVGugo39pa6rycEZLjtRPDfW84UAx',
    },
  },
  {
    keyName: 'LSP3IssuedAssets[]',
    value: [
      '0xD94353D9B005B3c0A9Da169b768a31C57844e490',
      '0xDaea594E385Fc724449E3118B2Db7E86dFBa1826',
    ],
  },
  {
    keyName: 'LSP1UniversalReceiverDelegate',
    value: '0x1183790f29BE3cDfD0A102862fEA1a4a30b3AdAb',
  },
]);
/**
{
  keys: [
    '0x5ef83ad95590333e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
    '0x3a47ab5bd3a594c3a8995f8fa58d0876c96819ca4516bd76100c92462f2f9dc0',
    '0x3a47ab5bd3a594c3a8995f8fa58d0876000000000000000000000000000000000000',
    '0x3a47ab5bd3a594c3a8995f8fa58d0876000000000000000000000000000000000001',
    '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47'
  ],
  values: [

'0x6f357c6a820464ddfac1bec070cc14a8daf04129871d458f2ca94368aae8391311af6361697066733a2f2f516d597231564a4c776572673670456f736364685647756676f3339706136727963455a4c6a745250446657383455417 8',
    '0x0000000000000000000000000000000000000000000000000000000000000002',
    '0xd94353d9b005b3c0a9da169b768a31c57844e490',
    '0xdaea594e385fc724449e3118b2db7e86dfba1826',
    '0x1183790f29be3cdfd0a102862fea1a4a30b3adab'
  ]
}
*/

console.log('/*-------------------------------------------------------*/');
console.log('/* encodedData - one key (LSP3Profile) with JSON       */');
console.log('/*-------------------------------------------------------*/');
console.log(encodedDataOneKey);
console.log('/*-------------------------------------------------------*/');
console.log('/* encodedData - one key (LSP3Profile) with hash       */');
console.log('/*-------------------------------------------------------*/');
console.log(encodedDataOneKeyV2);
console.log('/*-------------------------------------------------------*/');
console.log('/* encodedData - many keys                             */');
```

```
console.log('/*----------------------------------------------------------*/');
console.log(encodedDataManyKeys);
```

</file>

<file>

path: /examples/src/fetchData.js

url: https://github.com/ERC725Alliance/erc725.js/blob/main/examples/src/fetchData.js

```
// https://stackoverflow.com/questions/60059121/nodejs-es6-imports-cannot-find-module#comment106219502_60059121
// eslint-disable-next-line import/extensions
import { getInstance } from './instantiation.js';

const myERC725 = getInstance();

const dataAllKeys = await myERC725.fetchData();
/**
[
  {
    key: '0xeafec4d89fa9619884b600005ef83ad9559033e6e941db7d7c495acdce616347',
    name: 'SupportedStandards:LSP3Profile',
    value: false
  },
  {
    key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
    name: 'LSP3Profile',
    value: { LSP3Profile: [Object] }
  },
  {
    key: '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
    name: 'LSP1UniversalReceiverDelegate',
    value: '0x50A02EF693fF6961A7F9178d1e53CC8BbE1DaD68'
  },
  {
    key: '0x3a47ab5bd3a594c3a8995f8fa58d0876c96819ca4516bd76100c92462f2f9dc0',
    name: 'LSP3IssuedAssets[]',
    value: [
      '0xc444009d38d3046bb0cF81Fa2Cd295ce46A67C78',
      '0x4fEbC3491230571F6e1829E46602e3b110215A2E',
      '0xB92a8DdA2886384911AEE5C2a003D4CAbfa47aE3F',
      '0x1e52e7F1707dcda57dD33F003B2311652A465acA',
      '0x0BDA71aA980D37Ea56E8a3784E4c309101DAf3E4',
      '0xfDB4D9C299438B9839e9d04E34B9609C5b566000D',
      '0x081D3F0bff8ae2339cb65113822eEc1510704d5c',
      '0x55C98c6944B7497FaAf4db0386a1aD1E6efF526E',
      '0x90D1a1D68fa23AEEE991220703f1a1C3782e0b35',
      '0xdB5AB19792d9fB61c1Dff57810Fb7C6f839Af8ED'
    ]
```

```
  }
]
*/

const dataOneKey = await myERC725.fetchData('LSP3Profile');
/**
{
  key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
  name: 'LSP3Profile',
  value: {
    LSP3Profile: {
      name: 'patrick-mcdowell',
      links: [Array],
      description: "Beautiful clothing that doesn't cost the Earth. A sustainable designer based in London Patrick works with brand partners to refocus on systemic change centred around creative education. ",
      profileImage: [Array],
      backgroundImage: [Array],
      tags: [Array]
    }
  }
}
*/

const dataManyKeys = await myERC725.fetchData([
  'LSP3Profile',
  'LSP1UniversalReceiverDelegate',
]);
/**
[
  {
    key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
    name: 'LSP3Profile',
    value: { LSP3Profile: [Object] }
  },
  {
    key: '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
    name: 'LSP1UniversalReceiverDelegate',
    value: '0x50A02EF693fF6961A7F9178d1e53CC8BbE1DaD68'
  }
]
*/

console.log('/*-------------------------------------------*/');
console.log('/* fetchData - all keys                      */');
console.log('/*-------------------------------------------*/');
console.log(dataAllKeys);

console.log('/*-------------------------------------------*/');
console.log('/* fetchData - one key                       */');
console.log('/*-------------------------------------------*/');
console.log(dataOneKey);
```

```
console.log('/*-------------------------------------------/*');
console.log('/* fetchData - many keys                     /*');
console.log('/*-------------------------------------------/*');
console.log(dataManyKeys);
```

</file>

<file>

path: /examples/src/getData.js

url: https://github.com/ERC725Alliance/erc725.js/blob/main/examples/src/getData.js

```
// https://stackoverflow.com/questions/60059121/nodejs-es6-imports-cannot-find-module#comment106219502_60059121
// eslint-disable-next-line import/extensions
import { getInstance } from './instantiation.js';

const myERC725 = getInstance();

const dataAllKeys = await myERC725.getData();
/**
[
 {
   name: 'LSP3Profile',
   key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
   value: {
     verification: {
       method: 'keccak256(utf8)',
       data: '0xd96ff7776660095f661d16010c4349aa7478a9129ce0670f771596a6ff2d864a',
     },
     url: 'ipfs://QmbTmcbp8ZW23vkQrqkasMFqNg2z1iP4e3BCUMz9PKDsSV'
   }
 },
 {
   key: '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
   name: 'LSP1UniversalReceiverDelegate',
   value: '0x50A02EF693fF6961A7F9178d1e53CC8BbE1DaD68'
 },
 {
   name: 'LSP3IssuedAssets[]',
   key: '0x3a47ab5bd3a594c3a8995f8fa58d0876c96819ca4516bd76100c92462f2f9dc0',
   value: [
     '0xc444009d38d3046bb0cF81Fa2Cd295ce46A67C78',
     '0x4fEbC3491230571F6e1829E46602e3b110215A2E',
     '0xB92a8DdA2886384911AEE5C2a003D4CAbfa47aE3F',
     '0x1e52e7F1707dcda57dD33F003B2311652A465acA',
     '0x0BDA71aA980D37Ea56E8a3784E4c309101DAf3E4',
     '0xfDB4D9C299438B9839e9d04E34B9609C5b566000D',
     '0x081D3F0bff8ae2339cb65113822eEc1510704d5c',
     '0x55C98c6944B7497FaAf4db0386a1aD1E6efF526E',
```

```
    '0x90D1a1D68fa23AEEE991220703f1a1C3782e0b35',
    '0xdB5AB19792d9fB61c1Dff57810Fb7C6f839Af8ED'
   ]
  }
]
*/

const dataOneKey = await myERC725.getData('LSP3Profile');
/**
{
  key: '0x5ef83ad95590303e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
  name: 'LSP3Profile',
  value: {
    verification: {
      method: 'keccak256(utf8)',
      data: '0xd96ff7776660095f661d16010c4349aa7478a9129ce0670f771596a6ff2d864a',
    },
    url: 'ipfs://QmbTmcbp8ZW23vkQrqkasMFqNg2z1iP4e3BCUMz9PKDsSV'
  }
}
*/

const dataManyKeys = await myERC725.getData([
  'LSP3Profile',
  'LSP1UniversalReceiverDelegate',
]);
/**
[
  {
    key: '0x5ef83ad95590303e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
    name: 'LSP3Profile',
    value: {
      verification: {
        method: 'keccak256(utf8)',
        data: '0xd96ff7776660095f661d16010c4349aa7478a9129ce0670f771596a6ff2d864a',
      },
      url: 'ipfs://QmbTmcbp8ZW23vkQrqkasMFqNg2z1iP4e3BCUMz9PKDsSV'
    }
  },
  {
    key: '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
    name: 'LSP1UniversalReceiverDelegate',
    value: '0x50A02EF693fF6961A7F9178d1e53CC8BbE1DaD68'
  }
]
*/

console.log('/*-------------------------------------------/*');
console.log('/* getData - all keys                        /*');
console.log('/*-------------------------------------------/*');
console.log(dataAllKeys);
```

```
console.log('/*-------------------------------------------*/');
console.log('/* getData - one key                         */');
console.log('/*-------------------------------------------*/');
console.log(dataOneKey);

console.log('/*-------------------------------------------*/');
console.log('/* getData - many keys                       */');
console.log('/*-------------------------------------------*/');
console.log(dataManyKeys);
```

</file>

<file>

path: /examples/src/instantiation.js

url: https://github.com/ERC725Alliance/erc725.js/blob/main/examples/src/instantiation.js

```
import Web3 from 'web3';
import { ERC725 } from '@erc725/erc725.js';

// this is needed because node does not support  fetch  out of the box
// isomorphic-fetch is not needed in a browser environment
import 'isomorphic-fetch';

const RPC_ENDPOINT = 'https://rpc.testnet.lukso.network';
const IPFS_GATEWAY = 'https://2eff.lukso.dev/ipfs/';

export function getInstance() {
  const schema = [
    {
      name: 'SupportedStandards:LSP3Profile',
      key: '0xeafec4d89fa9619884b600005ef83ad9559033e6e941db7d7c495acdce616347',
      keyType: 'Mapping',
      valueContent: '0x5ef83ad9',
      valueType: 'bytes',
    },
    {
      name: 'LSP3Profile',
      key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
      keyType: 'Singleton',
      valueContent: 'JSONURL',
      valueType: 'bytes',
    },
    {
      name: 'LSP1UniversalReceiverDelegate',
      key: '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
      keyType: 'Singleton',
      valueContent: 'Address',
      valueType: 'address',
    },
```

```
  {
    name: 'LSP3IssuedAssets[]',
    key: '0x3a47ab5bd3a594c3a8995f8fa58d0876c96819ca4516bd76100c92462f2f9dc0',
    keyType: 'Array',
    valueContent: 'Address',
    valueType: 'address',
  },
];

const address = '0x7b2C957209897bc4423162e57D8C3CA863DCfBCc';
const provider = new Web3.providers.HttpProvider(RPC_ENDPOINT);
const config = {
  ipfsGateway: IPFS_GATEWAY,
};

return new ERC725(schema, address, provider, config);
}

export const profileJson = {
  LSP3Profile: {
    name: 'frozeman',
    links: [
      { title: 'Twitter', url: 'https://twitter.com/feindura' },
      { title: 'lukso.network', url: 'https://lukso.network' },
    ],
    description: 'The inventor fo ERC725 and ERC20.....',
    profileImage: [
      {
        width: 1800,
        height: 1712,
        verification: {
          method: 'keccak256(bytes)',
          data: '0xfbcfcbbc86d886e8624193611c48251d778884a90429d36c8d002559cbcb52972',
        },
        url: 'ipfs://QmNPh6hP5igFzPf4mPtKBa6Wttnmi3YNVMAptC7drzyeDB',
      },
      {
        width: 1024,
        height: 974,
        verification: {
          method: 'keccak256(bytes)',
          data: '0xa9399df007997de92a820c6c2ec1cb2d3f5aa5fc1adf294157de563eba39bb6e',
        },
        url: 'ipfs://QmW4wM4r9yWeY1gUCtt7c6v3ve7Fzdg8CKvTS96NU9Uiwr',
      },
      {
        width: 640,
        height: 609,
        verification: {
          method: 'keccak256(bytes)',
          data: '0xb316a695125cb0566da252266cfc9d5750a740bbdffa86712bb175008e70e6a31',
        },
```

```
      url: 'ipfs://QmXGELsqGidAHMwYRsEv6Z4emzMggtc5GXZYGFK7r6zFBg',
    },
    {
      width: 320,
      height: 304,
      verification: {
        method: 'keccak256(bytes)',
        data: '0xd22a272ff5b257056cc302bcdcca1c0ea00bf912aef310eb3fa7556696b1e780',
      },
      url: 'ipfs://QmRr2urTVi12VzYa5cSHDjJXACfapaeGZW2BuNmQ8rHjCG',
    },
    {
      width: 180,
      height: 171,
      verification: {
        method: 'keccak256(bytes)',
        data: '0xf80d0c1492de5e148392b3a7247396682e9b9564ef5fb97c06c1574bbb5e5f340',
      },
      url: 'ipfs://QmVeFyhHtdXR34UZanqU2qSuBTfGtBraG7hhN5byjJNAY5',
    },
  ],
  backgroundImage: [
    {
      width: 1800,
      height: 1013,
      verification: {
        method: 'keccak256(bytes)',
        data: '0x98fe032f81c43426fbcfb21c780c879667a08e2a65e8ae38027d4d61cdfe6f55',
      },
      url: 'ipfs://QmPJESHbVkPtSaHntNVY5F6JDLW8v69M2d6khXEYGUMn7N',
    },
    {
      width: 1024,
      height: 576,
      verification: {
        method: 'keccak256(bytes)',
        data: '0xfce1c7436a77a009a97e48e4e10c92e89fd95fe1556fc5c62ecef57cea51aa37',
      },
      url: 'ipfs://QmZc9uMJxyUeUpuowJ7AD6MKoNTaWdVNcBj72iisRyM9Su',
    },
    {
      width: 640,
      height: 360,
      verification: {
        method: 'keccak256(bytes)',
        data: '0x10a5cf2479992f1c555ad71e0a28668277f66fef6941a0c99f8d3b03e6b8b4009',
      },
      url: 'ipfs://QmbP3eTmUx1UQ2eZ8hrDz8j98yP2CTmsJvfp72LZKnkKj1',
    },
  ],
  tags: ['public profile'],
},
```

```
};
```

</file>

<file>

path: /package.json

url: https://github.com/ERC725Alliance/erc725.js/blob/main/package.json

```
{
  "name": "@erc725/erc725.js",
  "version": "0.21.2",
  "description": "Library to interact with ERC725 smart contracts",
  "main": "build/main/src/index.js",
  "typings": "build/main/src/index.d.ts",
  "module": "build/module/src/index.js",
  "files": [
    "build",
    "schemas",
    "docs"
  ],
  "scripts": {
    "build": "run-p build:*",
    "build:main": "tsc -p tsconfig.json",
    "build:module": "tsc -p tsconfig.module.json",
    "test": "env TS_NODE_COMPILER_OPTIONS='{\"module\": \"commonjs\" }' nyc --reporter=text --
reporter=lcov mocha",
    "lint": "eslint . --ext .ts",
    "release": "standard-version"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/ERC725Alliance/erc725.js"
  },
  "keywords": [
    "ethereum",
    "erc725",
    "lsp"
  ],
  "contributors": [
    {
      "name": "Robert McLeod",
      "url": "https://github.com/robertdavid010"
    },
    {
      "name": "Fabian Vogelsteller",
      "url": "https://github.com/frozeman"
    },
    {
      "name": "Hugo Masclet",
```

```
    "url": "https://github.com/Hugoo"
  },
  {
    "name": "Callum Grindle",
    "url": "https://github.com/CallumGrindle"
  }
],
"license": "Apache-2.0",
"bugs": {
  "url": "https://github.com/ERC725Alliance/erc725.js/issues"
},
"homepage": "https://github.com/ERC725Alliance/erc725.js#readme",
"devDependencies": {
  "@types/chai": "^4.3.4",
  "@types/isomorphic-fetch": "^0.0.36",
  "@types/jest": "^27.5.2",
  "@types/mocha": "^10.0.1",
  "@types/node": "^18.11.19",
  "@types/sinon": "^10.0.13",
  "@typescript-eslint/eslint-plugin": "^5.50.0",
  "@typescript-eslint/parser": "^5.50.0",
  "assert": "^2.0.0",
  "chai": "^4.3.7",
  "eslint": "^8.33.0",
  "eslint-config-airbnb-base": "^15.0.0",
  "eslint-config-prettier": "^8.6.0",
  "eslint-plugin-import": "^2.27.5",
  "eslint-plugin-prettier": "^5.0.0",
  "isomorphic-fetch": "^3.0.0",
  "jest": "^27.5.1",
  "mocha": "^10.0.0",
  "npm-run-all": "^4.1.5",
  "nyc": "^15.1.0",
  "prettier": "^3.0.0",
  "sinon": "^15.0.1",
  "standard-version": "^9.5.0",
  "ts-node": "^10.9.1",
  "typescript": "^4.9.5",
  "web3": "^1.10.0"
},
"dependencies": {
  "add": "^2.0.6",
  "ethereumjs-util": "^7.1.5",
  "web3-eth-abi": "^1.10.0",
  "web3-providers-http": "^1.10.0",
  "web3-utils": "^1.10.0"
}
}
```


</file>

<file>

path: /schemas/LSP10ReceivedVaults.json

url: https://github.com/ERC725Alliance/erc725.js/blob/main/schemas/LSP10ReceivedVaults.json

```
[
 {
  "name": "LSP10VaultsMap:<address>",
  "key": "0x192448c3c0f88c7f238c0000<address>",
  "keyType": "Mapping",
  "valueType": "(bytes4,uint128)",
  "valueContent": "(Bytes4,Number)"
 },
 {
  "name": "LSP10Vaults[]",
  "key": "0x55482936e01da86729a45d2b87a6b1d3bc582bea0ec00e38bdb340e3af6f9f06",
  "keyType": "Array",
  "valueType": "address",
  "valueContent": "Address"
 }
]
```

</file>

<file>

path: /schemas/LSP12IssuedAssets.json

url: https://github.com/ERC725Alliance/erc725.js/blob/main/schemas/LSP12IssuedAssets.json

```
[
 {
  "name": "LSP12IssuedAssets[]",
  "key": "0x7c8c3416d6cda87cd42c71ea1843df28ac4850354f988d55ee2eaa47b6dc05cd",
  "keyType": "Array",
  "valueType": "address",
  "valueContent": "Address"
 },
 {
  "name": "LSP12IssuedAssetsMap:<address>",
  "key": "0x74ac2555c10b9349e78f0000<address>",
  "keyType": "Mapping",
  "valueType": "(bytes4,uint128)",
  "valueContent": "(Bytes4,Number)"
 }
]
```

</file>

<file>

path: /schemas/LSP17ContractExtension.json

url: https://github.com/ERC725Alliance/erc725.js/blob/main/schemas/LSP17ContractExtension.json

```
[
 {
  "name": "LSP17Extension:<bytes4>",
  "key": "0xcee78b4094da860110960000<bytes4>",
  "keyType": "Mapping",
  "valueType": "address",
  "valueContent": "Address"
 }
]
```

</file>

<file>

path: /schemas/LSP1UniversalReceiverDelegate.json

url: https://github.com/ERC725Alliance/erc725.js/blob/main/schemas/LSP1UniversalReceiverDelegate.json

```
[
 {
  "name": "LSP1UniversalReceiverDelegate",
  "key": "0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47",
  "keyType": "Singleton",
  "valueType": "address",
  "valueContent": "Address"
 },
 {
  "name": "LSP1UniversalReceiverDelegate:<bytes32>",
  "key": "0x0cfc51aec37c55a4d0b10000<bytes32>",
  "keyType": "Mapping",
  "valueType": "address",
  "valueContent": "Address"
 }
]
```

</file>

<file>

path: /schemas/LSP3ProfileMetadata.json

url: https://github.com/ERC725Alliance/erc725.js/blob/main/schemas/LSP3ProfileMetadata.json

```
[
```

```
{
  "name": "SupportedStandards:LSP3Profile",
  "key": "0xeafec4d89fa9619884b600005ef83ad9559033e6e941db7d7c495acdce616347",
  "keyType": "Mapping",
  "valueType": "bytes4",
  "valueContent": "0x5ef83ad9"
},
{
  "name": "LSP3Profile",
  "key": "0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5",
  "keyType": "Singleton",
  "valueType": "bytes",
  "valueContent": "JSONURL"
},
{
  "name": "LSP12IssuedAssets[]",
  "key": "0x7c8c3416d6cda87cd42c71ea1843df28ac4850354f988d55ee2eaa47b6dc05cd",
  "keyType": "Array",
  "valueType": "address",
  "valueContent": "Address"
},
{
  "name": "LSP12IssuedAssetsMap:<address>",
  "key": "0x74ac2555c10b9349e78f0000<address>",
  "keyType": "Mapping",
  "valueType": "(bytes4,uint128)",
  "valueContent": "(Bytes4,Number)"
},
{
  "name": "LSP5ReceivedAssets[]",
  "key": "0x6460ee3c0aac563ccbf76d6e1d07bada78e3a9514e6382b736ed3f478ab7b90b",
  "keyType": "Array",
  "valueType": "address",
  "valueContent": "Address"
},
{
  "name": "LSP5ReceivedAssetsMap:<address>",
  "key": "0x812c4334633eb816c80d0000<address>",
  "keyType": "Mapping",
  "valueType": "(bytes4,uint128)",
  "valueContent": "(Bytes4,Number)"
},
{
  "name": "LSP1UniversalReceiverDelegate",
  "key": "0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47",
  "keyType": "Singleton",
  "valueType": "address",
  "valueContent": "Address"
},
{
  "name": "LSP1UniversalReceiverDelegate:<bytes32>",
  "key": "0x0cfc51aec37c55a4d0b10000<bytes32>",
```

```
  "keyType": "Mapping",
  "valueType": "address",
  "valueContent": "Address"
 },
 {
  "name": "LSP17Extension:<bytes4>",
  "key": "0xcee78b4094da860110960000<bytes4>",
  "keyType": "Mapping",
  "valueType": "address",
  "valueContent": "Address"
 }
]
```

</file>

<file>

path: /schemas/LSP4DigitalAsset.json

url: https://github.com/ERC725Alliance/erc725.js/blob/main/schemas/LSP4DigitalAsset.json

```
[
 {
  "name": "SupportedStandards:LSP4DigitalAsset",
  "key": "0xeafec4d89fa9619884b60000a4d96624a38f7ac2d8d9a604ecf07c12c77e480c",
  "keyType": "Mapping",
  "valueType": "bytes4",
  "valueContent": "0xa4d96624"
 },
 {
  "name": "LSP4TokenName",
  "key": "0xdeba1e292f8ba88238e10ab3c7f88bd4be4fac56cad5194b6ecceaf653468af1",
  "keyType": "Singleton",
  "valueType": "string",
  "valueContent": "String"
 },
 {
  "name": "LSP4TokenSymbol",
  "key": "0x2f0a68ab07768e01943a599e73362a0e17a63a72e94dd2e384d2c1d4db932756",
  "keyType": "Singleton",
  "valueType": "string",
  "valueContent": "String"
 },
 {
  "name": "LSP4Metadata",
  "key": "0x9afb95cacc9f95858ec44aa8c3b685511002e30ae54415823f406128b85b238e",
  "keyType": "Singleton",
  "valueType": "bytes",
  "valueContent": "JSONURL"
 },
 {
  "name": "LSP4Creators[]",
  "key": "0x114bd03b3a46d48759680d81ebb2b414fda7d030a7105a851867accf1c2352e7",
  "keyType": "Array",
  "valueType": "address",
  "valueContent": "Address"
 },
 {
  "name": "LSP4CreatorsMap:<address>",
  "key": "0x6de85eaf5d982b4e5da00000<address>",
  "keyType": "Mapping",
  "valueType": "(bytes4,uint128)",
  "valueContent": "(Bytes4,Number)"
 }
]
```


</file>

<file>

path: /schemas/LSP4DigitalAssetLegacy.json

url: https://github.com/ERC725Alliance/erc725.js/blob/main/schemas/LSP4DigitalAssetLegacy.json


```
[
 {
  "name": "SupportedStandards:LSP4DigitalCertificate",
  "key": "0xeafec4d89fa9619884b6b89135626455000000000000000000000000abf0613c",
  "keyType": "Mapping",
  "valueContent": "0xabf0613c",
  "valueType": "bytes"
 }
]
```


</file>

<file>

path: /schemas/LSP5ReceivedAssets.json

url: https://github.com/ERC725Alliance/erc725.js/blob/main/schemas/LSP5ReceivedAssets.json


```
[
 {
  "name": "LSP5ReceivedAssets[]",
  "key": "0x6460ee3c0aac563ccbf76d6e1d07bada78e3a9514e6382b736ed3f478ab7b90b",
  "keyType": "Array",
  "valueType": "address",
  "valueContent": "Address"
 },
 {
  "name": "LSP5ReceivedAssetsMap:<address>",
  "key": "0x812c4334633eb816c80d0000<address>",
  "keyType": "Mapping",
  "valueType": "(bytes4,uint128)",
  "valueContent": "(Bytes4,Number)"
 }
]
```


</file>

<file>

path: /schemas/LSP6KeyManager.json

url: https://github.com/ERC725Alliance/erc725.js/blob/main/schemas/LSP6KeyManager.json

```
[
 {
  "name": "AddressPermissions[]",
  "key": "0xdf30dba06db6a30e65354d9a64c609861f089545ca58c6b4dbe31a5f338cb0e3",
  "keyType": "Array",
  "valueType": "address",
  "valueContent": "Address"
 },
 {
  "name": "AddressPermissions:Permissions:<address>",
  "key": "0x4b80742de2bf82acb3630000<address>",
  "keyType": "MappingWithGrouping",
  "valueType": "bytes32",
  "valueContent": "BitArray"
 },
 {
  "name": "AddressPermissions:AllowedCalls:<address>",
  "key": "0x4b80742de2bf393a64c70000<address>",
  "keyType": "MappingWithGrouping",
  "valueType": "(bytes4,address,bytes4,bytes4)[CompactBytesArray]",
  "valueContent": "(BitArray,Address,Bytes4,Bytes4)"
 },
 {
  "name": "AddressPermissions:AllowedERC725YDataKeys:<address>",
  "key": "0x4b80742de2bf866c29110000<address>",
  "keyType": "MappingWithGrouping",
  "valueType": "bytes[CompactBytesArray]",
  "valueContent": "Bytes"
 }
]
```


</file>

<file>

path: /schemas/LSP8IdentifiableDigitalAsset.json

url: https://github.com/ERC725Alliance/erc725.js/blob/main/schemas/LSP8IdentifiableDigitalAsset.json

```
[
 {
  "name": "LSP8TokenIdType",
  "key": "0x715f248956de7ce65e94d9d836bfead479f7e70d69b718d47bfe7b00e05b4fe4",
  "keyType": "Singleton",
  "valueType": "uint256",
  "valueContent": "Number"
 },
 {
  "name": "LSP8MetadataTokenURI:<address|uint256|bytes32|string>",
  "key": "0x1339e76a390b7b9ec9010000<address|uint256|bytes32|string>",
  "keyType": "Mapping",
  "valueType": "(bytes4,string)",
  "valueContent": "(Bytes4,URI)"
 },
 {
  "name": "LSP8TokenMetadataBaseURI",
  "key": "0x1a7628600c3bac7101f53697f48df381ddc36b9015e7d7c9c5633d1252aa2843",
  "keyType": "Singleton",
  "valueType": "(bytes4,string)",
  "valueContent": "(Bytes4,URI)"
 },
 {
  "name": "LSP8ReferenceContract",
  "key": "0x708e7b881795f2e6b6c2752108c177ec89248458de3bf69d0d43480b3e5034e6",
  "keyType": "Singleton",
  "valueType": "(address,bytes32)",
  "valueContent": "(Address,bytes32)"
 }
]
```


</file>

<file>

path: /schemas/LSP9Vault.json

url: https://github.com/ERC725Alliance/erc725.js/blob/main/schemas/LSP9Vault.json

```
[
 {
  "name": "SupportedStandards:LSP9Vault",
  "key": "0xeafec4d89fa9619884b600007c0334a14085fefa8b51ae5a40895018882bdb90",
  "keyType": "Mapping",
  "valueType": "bytes4",
  "valueContent": "0x7c0334a1"
 },
 {
  "name": "LSP1UniversalReceiverDelegate",
  "key": "0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47",
  "keyType": "Singleton",
  "valueType": "address",
  "valueContent": "Address"
 },
 {
  "name": "LSP1UniversalReceiverDelegate:<bytes32>",
  "key": "0x0cfc51aec37c55a4d0b10000<bytes32>",
  "keyType": "Mapping",
  "valueType": "address",
  "valueContent": "Address"
 },
 {
  "name": "LSP17Extension:<bytes4>",
  "key": "0xcee78b4094da860110960000<bytes4>",
  "keyType": "Mapping",
  "valueType": "address",
  "valueContent": "Address"
 }
]
```


</file>

<file>

path: /src/constants/constants.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/constants/constants.ts

```
/*
  This file is part of @erc725/erc725.js.
  @erc725/erc725.js is free software: you can redistribute it and/or modify
  it under the terms of the GNU Lesser General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.
  @erc725/erc725.js is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU Lesser General Public License for more details.
  You should have received a copy of the GNU Lesser General Public License
  along with @erc725/erc725.js.  If not, see <http://www.gnu.org/licenses/>.
```

```
*/

/* eslint-disable @typescript-eslint/ban-types */
import { numberToHex, keccak256 } from 'web3-utils';

import { MethodData, Encoding, Method } from '../types/Method';

// https://github.com/ERC725Alliance/ERC725/blob/develop/docs/ERC-725.md#specification
export const ERC725Y_INTERFACE_IDS = {
  // interface functions:
  //     - getData(bytes32)
  //     - setData(bytes32,bytes)
  legacy: '0x2bd57b73',
  // interface functions:
  //     - getData(bytes32[])
  //     - setData(bytes32[],bytes[])
  '2.0': '0x5a988c0f',
  // version 3.0.0 introduced function overloading
  // interface functions:
  //     - getData(bytes32)
  //     - setData(bytes32,bytes)
  //     - getData(bytes32[])
  //     - setData(bytes32[],bytes[])
  '3.0': '0x714df77c',
  // InterfaceId of version 3 == interfaceId of version 4
  // version 5.0.0 removed function overloading
  // interface functions:
  //     - getData(bytes32)
  //     - setData(bytes32,bytes)
  //     - getDataBatch(bytes32[])
  //     - setDataBatch(bytes32[],bytes[])
  '5.0': '0x629aa694',
};

export enum ERC725_VERSION {
  NOT_ERC725 = 'NOT_ERC725',
  // The ERC725Y_LEGACY version uses getData(bytes32) function
  ERC725_LEGACY = 'ERC725_LEGACY',
  // The ERC725_v2 version uses getData(bytes32[]) function, as well as v3 and v4
  ERC725_v2 = 'ERC725_v2', // https://github.com/ERC725Alliance/ERC725/releases/tag/v2.2.0
  // The ERC725_v5 version uses getDataBatch(bytes32[]) function
  ERC725_v5 = 'ERC725_v5', // https://github.com/ERC725Alliance/ERC725/releases/tag/v5.0.0
}

export const METHODS: Record<Method, MethodData> = {
  [Method.GET_DATA_LEGACY]: {
    // Legacy version of ERC725Y - before v0.3.0
    sig: '0x54f6127f',
    value: numberToHex(0),
    returnEncoding: Encoding.BYTES,
  },
  [Method.GET_DATA]: {
```

```
    // https://github.com/ERC725Alliance/ERC725/blob/v4.0.0/docs/ERC-725.md#erc725y
    sig: '0x4e3e6e9c',
    value: numberToHex(0),
    returnEncoding: Encoding.BYTES_ARRAY,
  },
  [Method.GET_DATA_BATCH]: {
    // https://github.com/ERC725Alliance/ERC725/blob/v5.1.0/docs/ERC-725.md#erc725y
    sig: '0xdedff9c6',
    value: numberToHex(0),
    returnEncoding: Encoding.BYTES_ARRAY,
  },
  [Method.OWNER]: {
    sig: '0x8da5cb5b',
    value: numberToHex(0),
    returnEncoding: Encoding.ADDRESS,
  },
  [Method.SUPPORTS_INTERFACE]: {
    // https://eips.ethereum.org/EIPS/eip-165
    sig: '0x01ffc9a7',
    value: numberToHex(0),
    returnEncoding: Encoding.BOOL,
  },
  [Method.IS_VALID_SIGNATURE]: {
    // https://eips.ethereum.org/EIPS/eip-1271
    sig: '0x1626ba7e',
    value: numberToHex(0),
    returnEncoding: Encoding.BYTES4,
  },
};

export const UNKNOWN_VERIFICATION_METHOD = 'unknown';

export enum SUPPORTED_VERIFICATION_METHOD_STRINGS {
  KECCAK256_UTF8 = 'keccak256(utf8)',
  KECCAK256_BYTES = 'keccak256(bytes)',
}

export enum SUPPORTED_VERIFICATION_METHOD_HASHES {
  HASH_KECCAK256_UTF8 = '0x6f357c6a',
  HASH_KECCAK256_BYTES = '0x8019f9b1',
}

export type SUPPORTED_VERIFICATION_METHODS =
  | SUPPORTED_VERIFICATION_METHOD_STRINGS
  | SUPPORTED_VERIFICATION_METHOD_HASHES;

export const SUPPORTED_VERIFICATION_METHODS_LIST = Object.values(
  SUPPORTED_VERIFICATION_METHOD_STRINGS,
);

function keccak256Utf8(data) {
  return keccak256(JSON.stringify(data));
```

```
}

const KECCAK256_UTF8 = {
  method: keccak256Utf8,
  name: SUPPORTED_VERIFICATION_METHOD_STRINGS.KECCAK256_UTF8,
  sig: SUPPORTED_VERIFICATION_METHOD_HASHES.HASH_KECCAK256_UTF8,
};

const KECCAK256_BYTES = {
  method: keccak256,
  name: SUPPORTED_VERIFICATION_METHOD_STRINGS.KECCAK256_BYTES,
  sig: SUPPORTED_VERIFICATION_METHOD_HASHES.HASH_KECCAK256_BYTES,
};

export const HASH_METHODS: {
  [key: string]: {
    method: Function;
    name: SUPPORTED_VERIFICATION_METHOD_STRINGS;
    sig: SUPPORTED_VERIFICATION_METHODS;
  };
} = {
  [SUPPORTED_VERIFICATION_METHOD_STRINGS.KECCAK256_UTF8]: KECCAK256_UTF8,
  [SUPPORTED_VERIFICATION_METHOD_HASHES.HASH_KECCAK256_UTF8]: KECCAK256_UTF8,
  [SUPPORTED_VERIFICATION_METHOD_STRINGS.KECCAK256_BYTES]: KECCAK256_BYTES,
  [SUPPORTED_VERIFICATION_METHOD_HASHES.HASH_KECCAK256_BYTES]:
KECCAK256_BYTES,
};

// TODO: These values can be imported from lsp-smartcontracts lib after release
// prettier-ignore
export const LSP6_DEFAULT_PERMISSIONS = {
  CHANGEOWNER                  :
"0x0000000000000000000000000000000000000000000000000000000000000001",
  ADDCONTROLLER                :
"0x0000000000000000000000000000000000000000000000000000000000000002",
  EDITPERMISSIONS              :
"0x0000000000000000000000000000000000000000000000000000000000000004",
  ADDEXTENSIONS                :
"0x0000000000000000000000000000000000000000000000000000000000000008",
  CHANGEEXTENSIONS             :
"0x0000000000000000000000000000000000000000000000000000000000000010",
  ADDUNIVERSALRECEIVERDELEGATE :
"0x0000000000000000000000000000000000000000000000000000000000000020",
  CHANGEUNIVERSALRECEIVERDELEGATE :
"0x0000000000000000000000000000000000000000000000000000000000000040",
  REENTRANCY                   :
"0x0000000000000000000000000000000000000000000000000000000000000080",
  SUPER_TRANSFERVALUE          :
"0x0000000000000000000000000000000000000000000000000000000000000100",
  TRANSFERVALUE                :
"0x0000000000000000000000000000000000000000000000000000000000000200",
  SUPER_CALL                   :
```

```
"0x0000000000000000000000000000000000000000000000000000000000000400",
  CALL                        :
"0x0000000000000000000000000000000000000000000000000000000000000800",
  SUPER_STATICCALL            :
"0x0000000000000000000000000000000000000000000000000000000000001000",
  STATICCALL                  :
"0x0000000000000000000000000000000000000000000000000000000000002000",
  SUPER_DELEGATECALL          :
"0x0000000000000000000000000000000000000000000000000000000000004000",
  DELEGATECALL                :
"0x0000000000000000000000000000000000000000000000000000000000008000",
  DEPLOY                      :
"0x0000000000000000000000000000000000000000000000000000000000010000",
  SUPER_SETDATA               :
"0x0000000000000000000000000000000000000000000000000000000000020000",
  SETDATA                     :
"0x0000000000000000000000000000000000000000000000000000000000040000",
  ENCRYPT                     :
"0x0000000000000000000000000000000000000000000000000000000000080000",
  DECRYPT                     :
"0x0000000000000000000000000000000000000000000000000000000000100000",
  SIGN                        :
"0x0000000000000000000000000000000000000000000000000000000000200000",
  EXECUTE_RELAY_CALL          :
"0x0000000000000000000000000000000000000000000000000000000000400000",
};

export const LSP6_ALL_PERMISSIONS =
  '0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff';

export const COMPACT_BYTES_ARRAY_STRING = '[CompactBytesArray]';

export const DEFAULT_GAS_VALUE = 1_000_000;
```

</file>

<file>

path: /src/constants/interfaces.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/constants/interfaces.ts

```
/*
    This file is part of @erc725/erc725.js.
    @erc725/erc725.js is free software: you can redistribute it and/or modify
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    @erc725/erc725.js is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.
    You should have received a copy of the GNU Lesser General Public License
    along with @erc725/erc725.js.  If not, see <http://www.gnu.org/licenses/>.
*/

// from @lukso/lsp-smart-contracts v0.12.0, erc725.js should stay independent
export const INTERFACE_IDS_0_12_0 = {
  ERC1271: '0x1626ba7e',
  ERC725X: '0x7545acac',
  ERC725Y: '0x629aa694',
  LSP0ERC725Account: '0x24871b3d',
  LSP1UniversalReceiver: '0x6bb56a14',
  LSP1UniversalReceiverDelegate: '0xa245bbda',
  LSP6KeyManager: '0x23f34c62',
  LSP7DigitalAsset: '0xdaa746b7',
  LSP8IdentifiableDigitalAsset: '0x30dc5278',
  LSP9Vault: '0x28af17e6',
  LSP11BasicSocialRecovery: '0x049a28f1',
  LSP14Ownable2Step: '0x94be5999',
  LSP17Extendable: '0xa918fa6b',
  LSP17Extension: '0xcee78b40',
  LSP20CallVerification: '0x1a0eb6a5',
  LSP20CallVerifier: '0x0d6ecac7',
  LSP25ExecuteRelayCall: '0x5ac79908',
};

export interface AddressProviderOptions {
  address: string;
  provider: any;
}
```

</file>

<file>

path: /src/index.test.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/index.test.ts

```
/*
    This file is part of @erc725/erc725.js.
    @erc725/erc725.js is free software: you can redistribute it and/or modify
```

```
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    @erc725/erc725.js is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.
    You should have received a copy of the GNU Lesser General Public License
    along with web3.js.  If not, see <http://www.gnu.org/licenses/>.
*/
/**
 * @file test/test.ts
 * @author Robert McLeod <@robertdavid010>, Fabian Vogelsteller <fabian@lukso.network>
 * @date 2020
 */

// Tests for the @erc725/erc725.js package
import { assert } from 'chai';

import Web3 from 'web3';
import * as sinon from 'sinon';
import { hexToNumber, leftPad, numberToHex } from 'web3-utils';

import ERC725 from '.';
import {
  decodeKeyValue,
  encodeKey,
  encodeKeyValue,
  hashData,
} from './lib/utils';
import { ERC725JSONSchema } from './types/ERC725JSONSchema';
import { EthereumProvider, HttpProvider } from '../test/mockProviders';
import { mockSchema } from '../test/mockSchema';
import {
  generateAllData,
  generateAllRawData,
  generateAllResults,
} from '../test/testHelpers';

import 'isomorphic-fetch';

import {
  ERC725Y_INTERFACE_IDS,
  SUPPORTED_VERIFICATION_METHOD_STRINGS,
} from './constants/constants';
import { decodeKey } from './lib/decodeData';
import { INTERFACE_IDS_0_12_0 } from './constants/interfaces';

const address = '0x0c03fba782b07bcf810deb3b7f0595024a444f4e';

describe('Running @erc725/erc725.js tests...', () => {
  it('should throw when no arguments are supplied', () => {
```

```
    assert.throws(() => {
      // @ts-ignore
      // eslint-disable-next-line no-new
      new ERC725();
    }, 'Missing schema.');
  });

  it('should throw when incorrect or unsupported provider is provided', () => {
    assert.throws(() => {
      // @ts-ignore
      // eslint-disable-next-line no-new
      new ERC725(mockSchema, address, { test: false });
    }, /Incorrect or unsupported provider/);
  });

  it('should throw when calling getData without address & provider options set', async () => {
    const erc725 = new ERC725(mockSchema);
    try {
      await erc725.getData('LSP3Profile');
    } catch (error: any) {
      assert.deepStrictEqual(error.message, 'Missing ERC725 contract address.');
    }

    try {
      erc725.options.address = address;
      await erc725.getData('LSP3Profile');
    } catch (error: any) {
      assert.deepStrictEqual(error.message, 'Missing provider.');
    }
  });

  describe('isValidSignature', () => {
    it('should return true if the signature is valid [using rpcUrl]', async () => {
      const rpcUrl = 'https://rpc.l14.lukso.network';
      const erc725 = new ERC725(
        [],
        '0xD295E4748c1DFDFE028D7Dd2FEC3e52de2b1EB42', // result is mocked so we can use any
address
        rpcUrl,
      );
      const res = await erc725.isValidSignature(
        'hello',

'0x6c54ad4814ed6de85b9786e79de48ad0d597a243158194fa6b3604254ff58f9c2e4ffcc080e18a68c8e813f7
20b893c8d47d6f757b9e288a5814263642811c1b1c',
      );
      assert.deepStrictEqual(res, true);
    });

    it('should return true if the signature is valid [mock HttpProvider]', async () => {
      const provider = new HttpProvider({ returnData: [] }, [], true); // we mock a valid return response (magic
number)
```

```
    const erc725 = new ERC725(
      [],
      '0xD295E4748c1DFDFE028D7Dd2FEC3e52de2b1EB42', // result is mocked so we can use any
address
      provider,
    );

    const res = await erc725.isValidSignature(
      'hello',

'0x6c54ad4814ed6de85b9786e79de48ad0d597a243158194fa6b3604254ff58f9c2e4ffcc080e18a68c8e813f7
20b893c8d47d6f757b9e288a5814263642811c1b1c',
    );

    assert.deepStrictEqual(res, true);
  });

  it('should return true if the signature is valid [mock EthereumProvider]', async () => {
    const provider = new EthereumProvider({ returnData: [] }, [], true); // we mock a valid return response
(magic number)
    const erc725 = new ERC725(
      [],
      '0xD295E4748c1DFDFE028D7Dd2FEC3e52de2b1EB42', // result is mocked so we can use any
address
      provider,
    );

    const res = await erc725.isValidSignature(
      'hello',

'0x6c54ad4814ed6de85b9786e79de48ad0d597a243158194fa6b3604254ff58f9c2e4ffcc080e18a68c8e813f7
20b893c8d47d6f757b9e288a5814263642811c1b1c',
    );

    assert.deepStrictEqual(res, true);
  });

  it('should return false if the signature is invalid [using rpcUrl]', async () => {
    const contractAddress = '0xD295E4748c1DFDFE028D7Dd2FEC3e52de2b1EB42';
    const rpcUrl = 'https://rpc.l14.lukso.network';
    const erc725 = new ERC725(
      [],
      contractAddress, // result is mocked so we can use any address
      rpcUrl,
    );

    try {
      await erc725.isValidSignature(
        'wrong message',

'0x6c54ad4814ed6de85b9786e79de48ad0d597a243158194fa6b3604254ff58f9c2e4ffcc080e18a68c8e813f7
20b893c8d47d6f757b9e288a5814263642811c1b1c',
```

```
      );
      // should not reach this line
      assert.deepStrictEqual(true, false);
    } catch (error: any) {
      assert.deepStrictEqual(
        error.message,
         Error when checking signature. Is ${contractAddress} a valid contract address which supports
EIP-1271 standard? ,
      );
    }
  });

  it('should return false if the signature is valid [mock EthereumProvider]', async () => {
    const provider = new EthereumProvider({ returnData: [] }, [], false); // we mock a valid return response
    const erc725 = new ERC725(
      [],
      '0xD295E4748c1DFDFE028D7Dd2FEC3e52de2b1EB42', // result is mocked so we can use any
address
      provider,
    );

    const res = await erc725.isValidSignature(
      'hello',

'0xcafecafecafecafecafe6ce85b786ef79de48a43158194fa6b3604254ff58f9c2e4ffcc080e18a68c8e813f720b
893c8d47d6f757b9e288a5814263642811c1b1c',
    );

    assert.deepStrictEqual(res, false);
  });
});

describe('Getting all data in schema by provider [e2e]', () => {
  const web3 = new Web3('https://rpc.l14.lukso.network');

  const LEGACY_ERC725_CONTRACT_ADDRESS =
    '0xb8E120e7e5EAe7bfA629Db5CEFfA69C834F74e99';
  const ERC725_CONTRACT_ADDRESS =
    '0x320e678bEb3369702EA14555a74414B2C531c510';

  it('should return null if the key does not exist in the contract', async () => {
    const erc725 = new ERC725(
      [
        {
          name: 'ThisKeyDoesNotExist',
          key: '0xb12a0af5f83066646eb63c96bf29dcb827024d9a33189f5a61652a03951d1fbe',
          keyType: 'Singleton',
          valueContent: 'String',
          valueType: 'string',
        },
      ],
      ERC725_CONTRACT_ADDRESS,
```

```
    web3.currentProvider,
  );

  const data = await erc725.getData('ThisKeyDoesNotExist');

  const expectedResult = {
    name: 'ThisKeyDoesNotExist',
    key: '0xb12a0af5f83066646eb63c96bf29dcb827024d9a33189f5a61652a03951d1fbe',
    value: null,
  };

  assert.deepStrictEqual(data, expectedResult);

  const dataArray = await erc725.getData(['ThisKeyDoesNotExist']);
  assert.deepStrictEqual(dataArray, [expectedResult]);
});

it('should return [] if the key of type Array does not exist in the contract', async () => {
  const erc725 = new ERC725(
    [
      {
        name: 'NonExistingArray[]',
        key: '0xd6cbdbfc8d25c9ce4720b5fe6fa8fc536803944271617bf5425b4bd579195840',
        keyType: 'Array',
        valueContent: 'Address',
        valueType: 'address',
      },
    ],
    ERC725_CONTRACT_ADDRESS,
    web3.currentProvider,
  );

  const data = await erc725.getData('NonExistingArray[]');
  assert.deepStrictEqual(data, {
    name: 'NonExistingArray[]',
    key: '0xd6cbdbfc8d25c9ce4720b5fe6fa8fc536803944271617bf5425b4bd579195840',
    value: [],
  });

  const dataArray = await erc725.getData(['NonExistingArray[]']);
  assert.deepStrictEqual(dataArray, [
    {
      name: 'NonExistingArray[]',
      key: '0xd6cbdbfc8d25c9ce4720b5fe6fa8fc536803944271617bf5425b4bd579195840',
      value: [],
    },
  ]);
});

const e2eSchema: ERC725JSONSchema[] = [
  {
    name: 'LSP3Profile',
```

```
    key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
    keyType: 'Singleton',
    valueContent: 'JSONURL',
    valueType: 'bytes',
  },
  {
    name: 'LSP1UniversalReceiverDelegate',
    key: '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
    keyType: 'Singleton',
    valueContent: 'Address',
    valueType: 'address',
  },
];

const e2eResults = [
  {
    name: 'LSP3Profile',
    key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
    value: {
      verification: {
        method: 'keccak256(utf8)',
        data: '0x70546a2accab18748420b63c63b5af4cf710848ae83afc0c51dd8ad17fb5e8b3',
      },
      url: 'ipfs://QmecrGejUQVXpW4zS948pNvcnQrJ1KiAoM6bdfrVcWZsn5',
    },
  },
  {
    name: 'LSP1UniversalReceiverDelegate',
    key: '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
    value: '0x36e4Eb6Ee168EF54B1E8e850ACBE51045214B313',
  },
];

it('with web3.currentProvider [legacy]', async () => {
  const erc725 = new ERC725(
    e2eSchema,
    LEGACY_ERC725_CONTRACT_ADDRESS,
    web3.currentProvider,
  );
  const result = await erc725.getData();
  assert.deepStrictEqual(result, e2eResults);
});

it('with web3.currentProvider', async () => {
  const erc725 = new ERC725(
    e2eSchema,
    ERC725_CONTRACT_ADDRESS,
    web3.currentProvider,
  );
  const result = await erc725.getData();
  assert.deepStrictEqual(result, e2eResults);
});
```

```
});

describe('Getting data (using new getDataBatch) in schema', () => {
  const ERC725_V5_CONTRACT_ADDRESS =
    '0x4b30900F119E11D2A8CAe18176c4f9840E586Cc4';
  const web3 = new Web3('https://rpc.testnet.lukso.network');

  describe('By HttpProvider', () => {
    const provider = new HttpProvider(
      {
        returnData: [
          {
            key: '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
            value:

'0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000001000000000000000000000000000000000000000000000000000000000000002000000000000000000000000000000000000000000000000000000000000001436e4Eb6Ee168EF54B1E8e850ACBE51045214B313000000000000000000000000',
          },
        ],
      },
      [ERC725Y_INTERFACE_IDS['5.0']],
    );

    it('with http provider [ERC725Y_BATCH]', async () => {
      const erc725 = new ERC725(
        [
          {
            name: 'LSP1UniversalReceiverDelegate',
            key: '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
            keyType: 'Singleton',
            valueContent: 'Address',
            valueType: 'address',
          },
        ],
        '0x24464DbA7e7781a21eD86133Ebe88Eb9C0762620', // result is mocked so we can use any
address
        provider,
      );

      const [result] = await erc725.getData();
      assert.deepStrictEqual(result, {
        name: 'LSP1UniversalReceiverDelegate',
        key: '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
        value: '0x36e4Eb6Ee168EF54B1E8e850ACBE51045214B313',
      });
    });
  });

  describe('By provider [e2e] - luksoTestnet', () => {
    const e2eSchema: ERC725JSONSchema[] = [
```

```
    {
      name: 'LSP3Profile',
      key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
      keyType: 'Singleton',
      valueContent: 'JSONURL',
      valueType: 'bytes',
    },
    {
      name: 'LSP1UniversalReceiverDelegate',
      key: '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
      keyType: 'Singleton',
      valueContent: 'Address',
      valueType: 'address',
    },
  ];

  const e2eResults = [
    {
      name: 'LSP3Profile',
      key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
      value: {
        verification: {
          method: 'keccak256(utf8)',
          data: '0x70546a2accab18748420b63c63b5af4cf710848ae83afc0c51dd8ad17fb5e8b3',
        },
        url: 'ipfs://QmecrGejUQVXpW4zS948pNvcnQrJ1KiAoM6bdfrVcWZsn5',
      },
    },
    {
      name: 'LSP1UniversalReceiverDelegate',
      key: '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
      value: '0x36e4Eb6Ee168EF54B1E8e850ACBE51045214B313',
    },
  ];

  it('with web3.currentProvider [ERC725Y_BATCH]', async () => {
    const erc725 = new ERC725(
      e2eSchema,
      ERC725_V5_CONTRACT_ADDRESS,
      web3.currentProvider,
    );
    const result = await erc725.getData();
    assert.deepStrictEqual(result, e2eResults);
  });
 });
});

describe('Get/fetch edge cases [mock]', () => {
 it('should return null if the JSONURL is not set [fetchData]', async () => {
  const provider = new HttpProvider(
    {
      returnData: [
```

```
        {
          key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
          value:
'0x00000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000000000000000000000000000000000000000',
        },
      ],
    },
    [ERC725Y_INTERFACE_IDS.legacy],
  );
  const erc725 = new ERC725(
    [
      {
        name: 'LSP3Profile',
        key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
        keyType: 'Singleton',
        valueContent: 'URL',
        valueType: 'bytes',
      },
    ],
    '0x24464DbA7e7781a21eD86133Ebe88Eb9C0762620', // result is mocked so we can use any address
    provider,
  );

  const data = await erc725.fetchData('LSP3Profile');
  assert.deepStrictEqual(data, {
    name: 'LSP3Profile',
    key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
    value: null,
  });
});

it('should getData with multiple kind of input', async () => {
  // "Manual test" which checks if it handles well multiple kind of keys
  const provider = new HttpProvider(
    {
      returnData: [
        {
          key: '0x48643a15ac5407a175674ab0f8c92df5ae90694dac72ebf0a058fb2599e3b06a', // MyURL
          value:
'0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000000100000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000000000000000000000000000000000000035697066733a2f2f516d6245724b6833466a73415236596a73546a485a4e6d364d63447036615274383246746737639414a4a765a626400000000000000000000000',
        },
        {
          key: '0x74ac2555c10b9349e78f0000b74a88c43bcf691bd7a851f6603cb1868f6fc147', // LSP12IssuedAssetsMap:b74a88C43BCf691bd7A851f6603cb1868f6fc147
          value:
```

```
'0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000014
1098603b193d276f5fa176cc02007b609f9dae6b0000000000000000000000',
          },
          {
            key: '0xeafec4d89fa9619884b600005ef83ad9559033e6e941db7d7c495acdce616347', //
SupportedStandards:LSP3Profile
            value:

'0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000045
ef83ad900000000000000000000000000000000000000000000000000000000',
          },
        ],
      },
      [ERC725Y_INTERFACE_IDS['3.0']],
    );
    const erc725 = new ERC725(
      [
        {
          name: 'MyURL',
          key: '0x48643a15ac5407a175674ab0f8c92df5ae90694dac72ebf0a058fb2599e3b06a',
          keyType: 'Singleton',
          valueContent: 'URL',
          valueType: 'bytes',
        },
        {
          name: 'SupportedStandards:LSP3Profile',
          key: '0xeafec4d89fa9619884b600005ef83ad9559033e6e941db7d7c495acdce616347',
          keyType: 'Mapping',
          valueContent: '0x5ef83ad9',
          valueType: 'bytes',
        },
        {
          name: 'LSP12IssuedAssetsMap:<address>',
          key: '0x74ac2555c10b9349e78f0000<address>',
          keyType: 'Mapping',
          valueContent: 'Address',
          valueType: 'address',
        },
      ],
      '0x24464DbA7e7781a21eD86133Ebe88Eb9C0762620', // result is mocked so we can use any address
      provider,
    );

    const data = await erc725.getData([
      'MyURL',
      {
        keyName: 'LSP12IssuedAssetsMap:<address>',
```

```
        dynamicKeyParts: '0xb74a88C43BCf691bd7A851f6603cb1868f6fc147',
      },
      'SupportedStandards:LSP3Profile',
    ]);
    assert.deepStrictEqual(data, [
      {
        key: '0x48643a15ac5407a175674ab0f8c92df5ae90694dac72ebf0a058fb2599e3b06a',
        name: 'MyURL',
        value: 'ipfs://QmbErKh3FjsAR6YjsTjHZNm6McDp6aRt82Ftcv9AJJvZbd',
      },
      {
        key: '0x74ac2555c10b9349e78f0000b74a88c43bcf691bd7a851f6603cb1868f6fc147',
        name: 'LSP12IssuedAssetsMap:b74a88C43BCf691bd7A851f6603cb1868f6fc147',
        value: '0x1098603B193d276f5fA176CC02007B609F9DAE6b',
      },
      {
        key: '0xeafec4d89fa9619884b600005ef83ad9559033e6e941db7d7c495acdce616347',
        name: 'SupportedStandards:LSP3Profile',
        value: '0x5ef83ad9',
      },
    ]);
  });
});

[
  { name: 'legacy', interface: ERC725Y_INTERFACE_IDS.legacy },
  { name: 'latest', interface: ERC725Y_INTERFACE_IDS['3.0'] },
].forEach((contractVersion) => {
  describe(&amp;nbsp;Getting all data in schema by provider [ERC725Y ${contractVersion.name}][mock]&amp;nbsp;,
() => {
    // Construct the full data and results
    const fullResults = generateAllResults(mockSchema);
    const allRawData = generateAllRawData(
      mockSchema,
      contractVersion.interface === ERC725Y_INTERFACE_IDS['3.0'],
    );

    it('with web3.currentProvider', async () => {
      const provider = new HttpProvider({ returnData: allRawData }, [
        contractVersion.interface,
      ]);
      const erc725 = new ERC725(mockSchema, address, provider);
      const result = await erc725.getData();
      assert.deepStrictEqual(result, fullResults);
    });

    it('with ethereumProvider EIP 1193', async () => {
      const provider = new EthereumProvider({ returnData: allRawData }, [
        contractVersion.interface,
      ]);
      const erc725 = new ERC725(mockSchema, address, provider);
      const result = await erc725.getData();
```

```
    assert.deepStrictEqual(result, fullResults);
  });

  const testJSONURLSchema: ERC725JSONSchema = {
    name: 'TestJSONURL',
    key: '0xd154e1e44d32870ff5ade9e8726fd06d0ed6c996f5946dabfdfd46aa6dd2ea99',
    keyType: 'Singleton',
    valueContent: 'JSONURL',
    valueType: 'bytes',
  };

  it('fetchData JSONURL', async () => {
    const provider = new HttpProvider(
      {
        returnData: allRawData.filter(
          (rawData) =>
            rawData.key ===
            '0xd154e1e44d32870ff5ade9e8726fd06d0ed6c996f5946dabfdfd46aa6dd2ea99',
        ),
      },
      [contractVersion.interface],
    );

    const erc725 = new ERC725([testJSONURLSchema], address, provider);

    const jsonString = &amp;nbsp;{"LSP3Profile":
{"profileImage":"ipfs://QmYo8yg4zzmdu26NSvtsoKeU5oVR6h2ohmoa2Cx5i91mPf","backgroundImage":"ipf
s://QmZF5pxDJcB8eVvCd74rsXBFXhWL3S1XR5tty2cy1a58Ew","description":"Beautiful clothing that
doesn't cost the Earth. A sustainable designer based in London Patrick works with brand partners to refocus
on systemic change centred around creative education. "}}&amp;nbsp;;

    const fetchStub = sinon.stub(global, 'fetch');
    fetchStub.onCall(0).returns(Promise.resolve(new Response(jsonString)));
    const result = await erc725.fetchData('TestJSONURL');
    fetchStub.restore();

    assert.deepStrictEqual(result, {
      key: testJSONURLSchema.key,
      name: testJSONURLSchema.name,
      value: JSON.parse(jsonString),
    });
  });

  it('fetchData JSONURL with custom config.ipfsGateway', async () => {
    const provider = new HttpProvider(
      {
        returnData: allRawData.filter(
          (rawData) =>
            rawData.key ===
            '0xd154e1e44d32870ff5ade9e8726fd06d0ed6c996f5946dabfdfd46aa6dd2ea99',
        ),
      },
```

```
        [contractVersion.interface],
      );

      const ipfsGateway = 'https://2eff.lukso.dev';

      const erc725 = new ERC725([testJSONURLSchema], address, provider, {
        ipfsGateway,
      });

      const jsonString = &amp;nbsp;{"LSP3Profile":
{"profileImage":"ipfs://QmYo8yg4zzmdu26NSvtsoKeU5oVR6h2ohmoa2Cx5i91mPf","backgroundImage":"ipf
s://QmZF5pxDJcB8eVvCd74rsXBFXhWL3S1XR5tty2cy1a58Ew","description":"Beautiful clothing that
doesn't cost the Earth. A sustainable designer based in London Patrick works with brand partners to refocus
on systemic change centred around creative education. "}}&amp;nbsp;;

      const fetchStub = sinon.stub(global, 'fetch');
      fetchStub.onCall(0).returns(Promise.resolve(new Response(jsonString)));
      const result = await erc725.fetchData('TestJSONURL');
      assert.deepStrictEqual(result, {
        key: testJSONURLSchema.key,
        name: testJSONURLSchema.name,
        value: JSON.parse(jsonString),
      });
      fetchStub.restore();

      assert.ok(
        fetchStub.calledWith(
          &amp;nbsp;${ipfsGateway}/ipfs/QmbErKh3FjsAR6YjsTjHZNm6McDp6aRt82Ftcv9AJJvZbd&amp;nbsp;, // this
value comes from the mockSchema
        ),
      );
    });

    if (contractVersion.interface === ERC725Y_INTERFACE_IDS['3.0']) {
      it('fetchData JSONURL with dynamic key', async () => {
        const provider = new HttpProvider(
          {
            returnData: [
              {
                key: '0x84b02f6e50a0a0819a4f0000cafecafecafecafecafecafecafecafecafecafe',
                value:

'0x00000000000000000000000000000000000000000000000000000000000000200000000000000000000000
0000000000000000000000000000000000000000000100000000000000000000000000000000000000000000000000
0000000000000000000020000000000000000000000000000000000000000000000000000000000000005
6f357c6a733e78f2fc4a3304c141e8424d02c9069fe08950c6514b27289ead8ef4faa49d697066733a2f2f516d
6245724b6833466a73415236596a73546a485a4e6d364d6344703661527438324674637639414a4a765a62
6400000000000000',
              },
            ],
          },
          [contractVersion.interface],
```

```
        );
        const erc725 = new ERC725(
          [
            {
              name: 'JSONForAddress:<address>',
              key: '0x84b02f6e50a0a0819a4f0000cafecafecafecafecafecafecafecafecafecafe',
              keyType: 'Singleton',
              valueContent: 'JSONURL',
              valueType: 'bytes',
            },
          ],
          address,
          provider,
        );

        const jsonString = &amp;nbsp;{"LSP3Profile":
{"profileImage":"ipfs://QmYo8yg4zzmdu26NSvtsoKeU5oVR6h2ohmoa2Cx5i91mPf","backgroundImage":"ipf
s://QmZF5pxDJcB8eVvCd74rsXBFXhWL3S1XR5tty2cy1a58Ew","description":"Beautiful clothing that
doesn't cost the Earth. A sustainable designer based in London Patrick works with brand partners to refocus
on systemic change centred around creative education. "}}&amp;nbsp;;

        const fetchStub = sinon.stub(global, 'fetch');
        fetchStub
          .onCall(0)
          .returns(Promise.resolve(new Response(jsonString)));
        const result = await erc725.fetchData({
          keyName: 'JSONForAddress:<address>',
          dynamicKeyParts: '0xcafecafecafecafecafecafecafecafecafecafe',
        });
        fetchStub.restore();
        assert.deepStrictEqual(result, {
          name: 'JSONForAddress:cafecafecafecafecafecafecafecafecafecafe',
          key: '0x84b02f6e50a0a0819a4f0000cafecafecafecafecafecafecafecafecafecafe',
          value: JSON.parse(jsonString),
        });
      });
    }

    if (contractVersion.interface === ERC725Y_INTERFACE_IDS.legacy) {
      it('fetchData AssetURL', async () => {
        const fetchStub = sinon.stub(global, 'fetch');
        fetchStub
          .onCall(0)
          .returns(Promise.resolve(new Response(new Uint8Array(5))));

        const provider = new HttpProvider(
          {
            returnData: [
              {
                key: '0xf18290c9b373d751e12c5ec807278267a807c35c3806255168bc48a85757ceee',
                value:
```

```
'0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000000598019f9b1c41589e7559804ea4a2080dad19d876a024ccb05117835447d72ce08c1d020ec697066733a2f2f516d596f387967347a7a6d647532364e537674736f4b6555356f56523668326f686d6f61324378356939316d506600000000000000000',
            },

            // Encoded value of:
            // {
            //   verification: {
            //     method: 'keccak256(bytes)', // 0x8019f9b1
            //     data: '0xc41589e7559804ea4a2080dad19d876a024ccb05117835447d72ce08c1d020ec',
            //   },
            //   url: 'ipfs://QmYo8yg4zzmdu26NSvtsoKeU5oVR6h2ohmoa2Cx5i91mPf',
            // },
          ],
        },
        [contractVersion.interface],
      );

      const erc725 = new ERC725(
        [
          {
            name: 'TestAssetURL',
            key: '0xf18290c9b373d751e12c5ec807278267a807c35c3806255168bc48a85757ceee',
            keyType: 'Singleton',
            valueContent: 'AssetURL',
            valueType: 'bytes',
          },
        ],
        address,
        provider,
      );
      const result = await erc725.fetchData('TestAssetURL');

      fetchStub.restore();

      assert.strictEqual(
        Object.prototype.toString.call(result.value),
        '[object Uint8Array]',
      );
    });
  }
 });
});

describe('Getting data by schema element by provider', () => {
  mockSchema.forEach((schemaElement) => {
    it(schemaElement.name + ' with web3.currentProvider', async () => {
      const returnRawData = generateAllRawData([schemaElement], false);
      const provider = new HttpProvider({ returnData: returnRawData }, [
        ERC725Y_INTERFACE_IDS.legacy,
      ]);
```

```
      const erc725 = new ERC725(mockSchema, address, provider);
      const result = await erc725.getData(
        schemaElement.dynamicKeyParts
          ? {
              keyName: schemaElement.key,
              dynamicKeyParts: schemaElement.dynamicKeyParts,
            }
          : schemaElement.key,
      );
      assert.deepStrictEqual(result, {
        name: schemaElement.name,
        key: schemaElement.key,
        value: schemaElement.expectedResult,
      });
    });

    it(schemaElement.name + ' with ethereumProvider EIP 1193', async () => {
      const returnRawData = generateAllRawData([schemaElement], false);
      const provider = new HttpProvider({ returnData: returnRawData }, [
        ERC725Y_INTERFACE_IDS.legacy,
      ]);
      const erc725 = new ERC725(mockSchema, address, provider);
      const result = await erc725.getData(
        schemaElement.dynamicKeyParts
          ? {
              keyName: schemaElement.key,
              dynamicKeyParts: schemaElement.dynamicKeyParts,
            }
          : schemaElement.key,
      );
      assert.deepStrictEqual(result, {
        name: schemaElement.name,
        key: schemaElement.key,
        value: schemaElement.expectedResult,
      });
    });
  });
});

describe('Testing utility encoding & decoding functions', () => {
  const allGraphData = generateAllData(mockSchema) as any;
  /* ************************************* */
  /* Testing encoding/decoding field by field */
  for (let index = 0; index < mockSchema.length; index++) {
    const schemaElement = mockSchema[index];

    // ARRAY type:
    if (schemaElement.keyType.toLowerCase() === 'array') {
      it('Encode data values in array: ' + schemaElement.name, async () => {
        const results: string[] = [];

        // Encode array loop
```

```
  for (let i = 0; i < schemaElement.expectedResult.length; i++) {
    if (i === 0) {
      // Push the array length into the first element of results array
      results.push(
        leftPad(numberToHex(schemaElement.expectedResult.length), 32),
      );
    }

    results.push(
      encodeKeyValue(
        schemaElement.valueContent,
        schemaElement.valueType,
        schemaElement.expectedResult[i],
        schemaElement.name,
      ) as string,
    );
  } // end for loop
  assert.deepStrictEqual(results, schemaElement.returnGraphData);
});

it('decodes data values in array: ' + schemaElement.name, async () => {
  const results: any[] = [];

  // decode array loop
  for (let i = 0; i < schemaElement.returnGraphData.length; i++) {
    const element = schemaElement.returnGraphData[i];

    try {
      // Fail silently with anything BUT the arrayLength key
      hexToNumber(element.value);
    } catch (error) {
      const result = decodeKeyValue(
        schemaElement.valueContent,
        schemaElement.valueType,
        element,
        schemaElement.name,
      );

      // Handle object types
      if (
        result &&
        typeof result === 'object' &&
        Object.keys(result).length > 0
      ) {
        const objResult = {};

        for (let j = 0; index < Object.keys(result).length; j++) {
          const key = Object.keys(result)[j];
          const e = result[key];
          objResult[key] = e;
        }
```

```
            results.push(objResult);
          } else {
            results.push(result);
          }
          assert.deepStrictEqual(results, schemaElement.expectedResult);
        }
      } // end for loop
    });

    it(&amp;nbsp;encodes all data values for keyType "Array" in: ${schemaElement.name}&amp;nbsp;, async () => {
      const data = schemaElement.expectedResult;
      const intendedResults = allGraphData.filter(
        (e) => e.key.slice(0, 34) === schemaElement.key.slice(0, 34),
      );
      // handle '0x'....
      // intendedResults = intendedResults.filter(e => e !== '0x' && e.value !== '0x')
      const results = encodeKey(schemaElement, data);
      assert.deepStrictEqual(results, intendedResults);
    });

    it(&amp;nbsp;decodes all data values for keyType "Array" in: ${schemaElement.name}&amp;nbsp;, async () => {
      const values = allGraphData.filter(
        (e) => e.key.slice(0, 34) === schemaElement.key.slice(0, 34),
      );
      const intendedResults = schemaElement.expectedResult;
      const results = decodeKey(schemaElement, values);
      assert.deepStrictEqual(results, intendedResults);
    });

    it(&amp;nbsp;encodes all data values for keyType "Array" in naked class instance:
${schemaElement.name}&amp;nbsp;, async () => {
      const data = schemaElement.expectedResult;

      const keyValuePairs = allGraphData.filter(
        (e) => e.key.slice(0, 34) === schemaElement.key.slice(0, 34),
      );

      const intendedResult: { keys: string[]; values: string[] } = {
        keys: [],
        values: [],
      };

      keyValuePairs.forEach((keyValuePair) => {
        intendedResult.keys.push(keyValuePair.key);
        intendedResult.values.push(keyValuePair.value);
      });

      const erc725 = new ERC725([schemaElement]);

      const results = erc725.encodeData([
        {
          keyName: schemaElement.name,
```

```
            value: data,
          },
        ]);
        assert.deepStrictEqual(results, intendedResult);
      });

      it(&amp;nbsp;decode all data values for keyType "Array" in naked class instance:
${schemaElement.name}&amp;nbsp;, async () => {
        const values = allGraphData.filter(
          (e) => e.key.slice(0, 34) === schemaElement.key.slice(0, 34),
        );
        const intendedResults = schemaElement.expectedResult;
        const erc725 = new ERC725([schemaElement]);
        const results = erc725.decodeData([
          {
            keyName: schemaElement.name,
            value: values,
          },
        ]);
        assert.deepStrictEqual(results[0].value, intendedResults);
      });
    } else {
      if (schemaElement.dynamicKeyParts) {
        // eslint-disable-next-line no-continue
        continue;
      }

      // SINGLETON type: This is not an array, assumed 'Singleton'
      it('encodes data value for: ' + schemaElement.name, async () => {
        const result = encodeKeyValue(
          schemaElement.valueContent,
          schemaElement.valueType,
          schemaElement.expectedResult,
          schemaElement.name,
        );
        assert.deepStrictEqual(result, schemaElement.returnGraphData);
      });

      it('decodes data value for: ' + schemaElement.name, async () => {
        const result = decodeKeyValue(
          schemaElement.valueContent,
          schemaElement.valueType,
          schemaElement.returnGraphData,
          schemaElement.name,
        );
        assert.deepStrictEqual(result, schemaElement.expectedResult);
      });

      it(&amp;nbsp;Encode data value from naked class instance for ${schemaElement.name}&amp;nbsp;, async () =>
{
        const erc725 = new ERC725([schemaElement]);
        const result = erc725.encodeData([
```

```
        {
          keyName: schemaElement.name,
          value: schemaElement.expectedResult,
        },
      ]);
      assert.deepStrictEqual(result, {
        keys: [schemaElement.key],
        values: [schemaElement.returnGraphData],
      });
    });

    it( Decode data value from naked class instance for ${schemaElement.name} , async () =>
{
      const erc725 = new ERC725([schemaElement]);
      const result = erc725.decodeData([
        {
          keyName: schemaElement.name,
          value: schemaElement.returnGraphData,
          dynamicKeyParts: schemaElement.dynamicKeyParts,
        },
      ]);
      assert.deepStrictEqual(result[0].value, schemaElement.expectedResult);
    });
  }
 }
});

it('should encode/decode JSON properly', () => {
  const schema: ERC725JSONSchema[] = [
    {
      name: 'LSP3Profile',
      key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
      keyType: 'Singleton',
      valueContent: 'JSONURL',
      valueType: 'bytes',
    },
  ];

  const myERC725 = new ERC725(schema);

  const json = {
    name: 'rryter',
    description: 'Web Developer located in Switzerland.',
    profileImage: [
      {
        width: 1350,
        height: 1800,
        verification: {
          method: 'keccak256(bytes)',
          data: '0x229b60ea5b58e1ab8e6f1063300be110bb4fa663ba75d3814d60104ac6b74497',
        },
        url: 'ipfs://Qmbv9j6iCDDYJ1NXHTZnNHDJ6qaaKkZsf79jhUMFAXcfDR',
```

```
    },
    {
      width: 768,
      height: 1024,
      verification: {
        method: 'keccak256(bytes)',
        data: '0x320db57770084f114988c8a94bcf219ca66c69421590466a45f382cd84995c2b',
      },
      url: 'ipfs://QmS4m2LmRpay7Jij4DCpvaW5zKZYy43ATZdRxUkUND6nG3',
    },
  ],
  backgroundImage: [
    {
      width: 1024,
      height: 768,
      verification: {
        method: 'keccak256(bytes)',
        data: '0xbe2d39fe1e0b1911155afc74010db3483528a2b645dea8fcf47bdc34147769be',
      },
      url: 'ipfs://QmQ6ujfKSc91F44KtMe6WRTSCXoSdCjomQUy8hCUxHMr28',
    },
    {
      width: 640,
      height: 480,
      verification: {
        method: 'keccak256(bytes)',
        data: '0xb115f2bf09994e79726db27a7b8d5a0de41a5b81d11b59b3038fa158718266ff',
      },
      url: 'ipfs://QmakaRZxJMMqwQFJY98J3wjbqYVDnaSZ9sEqBF9iMv3GNX',
    },
  ],
  tags: ['public profile'],
  links: [],
};

const encodedData = myERC725.encodeData([
  {
    keyName: 'LSP3Profile',
    value: {
      json,
      url: 'ipfs://QmbKvCVEePiDKxuouyty9bMsWBAxZDGr2jhxd4pLGLx95D',
    },
  },
]);

const decodedData = myERC725.decodeData([
  {
    keyName: 'LSP3Profile',
    value: encodedData.values[0],
  },
]);
```

```
  assert.deepStrictEqual(
    decodedData[0].value.url,
    'ipfs://QmbKvCVEePiDKxuouyty9bMsWBAxZDGr2jhxd4pLGLx95D',
  );
  assert.deepStrictEqual(
    decodedData[0].value.verification.data,
    hashData(json, SUPPORTED_VERIFICATION_METHOD_STRINGS.KECCAK256_UTF8),
  );
  assert.deepStrictEqual(
    decodedData[0].value.verification.method,
    SUPPORTED_VERIFICATION_METHOD_STRINGS.KECCAK256_UTF8,
  );
});

describe('permissions', () => {
  const testCases: { hex: string; permissions }[] = [
    {
      permissions: {
        CHANGEOWNER: true,
        ADDCONTROLLER: true,
        EDITPERMISSIONS: true,
        ADDEXTENSIONS: true,
        CHANGEEXTENSIONS: true,
        ADDUNIVERSALRECEIVERDELEGATE: true,
        CHANGEUNIVERSALRECEIVERDELEGATE: true,
        REENTRANCY: false,
        SUPER_TRANSFERVALUE: true,
        TRANSFERVALUE: true,
        SUPER_CALL: true,
        CALL: true,
        SUPER_STATICCALL: true,
        STATICCALL: true,
        SUPER_DELEGATECALL: false,
        DELEGATECALL: false,
        DEPLOY: true,
        SUPER_SETDATA: true,
        SETDATA: true,
        ENCRYPT: true,
        DECRYPT: true,
        SIGN: true,
        EXECUTE_RELAY_CALL: false,
      },
      hex: '0x0000000000000000000000000000000000000000000000000000000000003f3f7f',
    },
    {
      permissions: {
        CHANGEOWNER: false,
        ADDCONTROLLER: false,
        EDITPERMISSIONS: false,
        ADDEXTENSIONS: false,
        CHANGEEXTENSIONS: false,
        ADDUNIVERSALRECEIVERDELEGATE: false,
```

```
    CHANGEUNIVERSALRECEIVERDELEGATE: false,
    REENTRANCY: false,
    SUPER_TRANSFERVALUE: false,
    TRANSFERVALUE: false,
    SUPER_CALL: false,
    CALL: false,
    SUPER_STATICCALL: false,
    STATICCALL: false,
    SUPER_DELEGATECALL: false,
    DELEGATECALL: false,
    DEPLOY: false,
    SUPER_SETDATA: false,
    SETDATA: false,
    ENCRYPT: false,
    DECRYPT: false,
    SIGN: false,
    EXECUTE_RELAY_CALL: false,
  },
  hex: '0x0000000000000000000000000000000000000000000000000000000000000000',
},
{
  permissions: {
    CHANGEOWNER: false,
    ADDCONTROLLER: false,
    EDITPERMISSIONS: false,
    ADDEXTENSIONS: false,
    CHANGEEXTENSIONS: false,
    ADDUNIVERSALRECEIVERDELEGATE: false,
    CHANGEUNIVERSALRECEIVERDELEGATE: false,
    REENTRANCY: false,
    SUPER_TRANSFERVALUE: false,
    TRANSFERVALUE: true,
    SUPER_CALL: false,
    CALL: true,
    SUPER_STATICCALL: false,
    STATICCALL: false,
    SUPER_DELEGATECALL: false,
    DELEGATECALL: false,
    DEPLOY: false,
    SUPER_SETDATA: false,
    SETDATA: false,
    ENCRYPT: false,
    DECRYPT: false,
    SIGN: true,
    EXECUTE_RELAY_CALL: false,
  },
  hex: '0x0000000000000000000000000000000000000000000000000000000000200a00',
},
{
  permissions: {
    CHANGEOWNER: false,
    ADDCONTROLLER: false,
```

```
    EDITPERMISSIONS: false,
    ADDEXTENSIONS: false,
    CHANGEEXTENSIONS: false,
    ADDUNIVERSALRECEIVERDELEGATE: false,
    CHANGEUNIVERSALRECEIVERDELEGATE: false,
    REENTRANCY: false,
    SUPER_TRANSFERVALUE: false,
    TRANSFERVALUE: false,
    SUPER_CALL: false,
    CALL: true,
    SUPER_STATICCALL: false,
    STATICCALL: false,
    SUPER_DELEGATECALL: false,
    DELEGATECALL: false,
    DEPLOY: false,
    SUPER_SETDATA: false,
    SETDATA: true,
    ENCRYPT: false,
    DECRYPT: false,
    SIGN: false,
    EXECUTE_RELAY_CALL: false,
  },
  hex: '0x0000000000000000000000000000000000000000000000000000000000040800',
},
{
  permissions: {
    CHANGEOWNER: false,
    ADDCONTROLLER: false,
    EDITPERMISSIONS: true,
    ADDEXTENSIONS: false,
    CHANGEEXTENSIONS: false,
    ADDUNIVERSALRECEIVERDELEGATE: false,
    CHANGEUNIVERSALRECEIVERDELEGATE: false,
    REENTRANCY: false,
    SUPER_TRANSFERVALUE: false,
    TRANSFERVALUE: false,
    SUPER_CALL: false,
    CALL: false,
    SUPER_STATICCALL: false,
    STATICCALL: false,
    SUPER_DELEGATECALL: false,
    DELEGATECALL: false,
    DEPLOY: false,
    SUPER_SETDATA: false,
    SETDATA: true,
    ENCRYPT: false,
    DECRYPT: false,
    SIGN: false,
    EXECUTE_RELAY_CALL: false,
  },
  hex: '0x0000000000000000000000000000000000000000000000000000000000040004',
},
```

```
  {
    permissions: {
      CHANGEOWNER: false,
      ADDCONTROLLER: false,
      EDITPERMISSIONS: false,
      ADDEXTENSIONS: false,
      CHANGEEXTENSIONS: false,
      ADDUNIVERSALRECEIVERDELEGATE: false,
      CHANGEUNIVERSALRECEIVERDELEGATE: false,
      REENTRANCY: false,
      SUPER_TRANSFERVALUE: false,
      TRANSFERVALUE: true,
      SUPER_CALL: false,
      CALL: true,
      SUPER_STATICCALL: false,
      STATICCALL: false,
      SUPER_DELEGATECALL: false,
      DELEGATECALL: false,
      DEPLOY: false,
      SUPER_SETDATA: false,
      SETDATA: false,
      ENCRYPT: false,
      DECRYPT: false,
      SIGN: false,
      EXECUTE_RELAY_CALL: false,
    },
    hex: '0x0000000000000000000000000000000000000000000000000000000000000a00',
  },
];

const erc725Instance = new ERC725([]);

describe( encodePermissions , () => {
  testCases.forEach((testCase) => {
    it( Encodes ${testCase.hex} permission correctly , () => {
      assert.deepStrictEqual(
        ERC725.encodePermissions(testCase.permissions),
        testCase.hex,
      );
      assert.deepStrictEqual(
        erc725Instance.encodePermissions(testCase.permissions),
        testCase.hex,
      );
    });
  });

  it('Defaults permissions to false if not passed', () => {
    assert.deepStrictEqual(
      ERC725.encodePermissions({
        EDITPERMISSIONS: true,
        SETDATA: true,
      }),
```

```
      '0x0000000000000000000000000000000000000000000000000000000000040004',
    );
    assert.deepStrictEqual(
      erc725Instance.encodePermissions({
        EDITPERMISSIONS: true,
        SETDATA: true,
      }),
      '0x0000000000000000000000000000000000000000000000000000000000040004',
    );
  });
});

describe('decodePermissions', () => {
  testCases.forEach((testCase) => {
    it( Decodes ${testCase.hex} permission correctly , () => {
      assert.deepStrictEqual(
        ERC725.decodePermissions(testCase.hex),
        testCase.permissions,
      );
      assert.deepStrictEqual(
        erc725Instance.decodePermissions(testCase.hex),
        testCase.permissions,
      );
    });
  });
  it( Decodes 0xfff...fff admin permission correctly , () => {
    assert.deepStrictEqual(
      ERC725.decodePermissions(
        '0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff',
      ),
      {
        CHANGEOWNER: true,
        ADDCONTROLLER: true,
        EDITPERMISSIONS: true,
        ADDEXTENSIONS: true,
        CHANGEEXTENSIONS: true,
        ADDUNIVERSALRECEIVERDELEGATE: true,
        CHANGEUNIVERSALRECEIVERDELEGATE: true,
        REENTRANCY: true,
        SUPER_TRANSFERVALUE: true,
        TRANSFERVALUE: true,
        SUPER_CALL: true,
        CALL: true,
        SUPER_STATICCALL: true,
        STATICCALL: true,
        SUPER_DELEGATECALL: true,
        DELEGATECALL: true,
        DEPLOY: true,
        SUPER_SETDATA: true,
        SETDATA: true,
        ENCRYPT: true,
        DECRYPT: true,
```

```
          SIGN: true,
          EXECUTE_RELAY_CALL: true,
        },
      );
      assert.deepStrictEqual(
        erc725Instance.decodePermissions(
          '0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff',
        ),
        {
          CHANGEOWNER: true,
          ADDCONTROLLER: true,
          EDITPERMISSIONS: true,
          ADDEXTENSIONS: true,
          CHANGEEXTENSIONS: true,
          ADDUNIVERSALRECEIVERDELEGATE: true,
          CHANGEUNIVERSALRECEIVERDELEGATE: true,
          REENTRANCY: true,
          SUPER_TRANSFERVALUE: true,
          TRANSFERVALUE: true,
          SUPER_CALL: true,
          CALL: true,
          SUPER_STATICCALL: true,
          STATICCALL: true,
          SUPER_DELEGATECALL: true,
          DELEGATECALL: true,
          DEPLOY: true,
          SUPER_SETDATA: true,
          SETDATA: true,
          ENCRYPT: true,
          DECRYPT: true,
          SIGN: true,
          EXECUTE_RELAY_CALL: true,
        },
      );
    });
  });
});

describe('getSchema', () => {
  it('should find key in schema used for instantiation', async () => {
    const schema: ERC725JSONSchema = {
      name: 'InstantiationSchema',
      key: '0xdbc90d23b2e4ff291c111a658864f9723a77b8c1f22b707e51a686413948206d',
      keyType: 'Singleton',
      valueContent: 'JSONURL',
      valueType: 'bytes',
    };

    const erc725 = new ERC725([schema]);

    const foundSchema = erc725.getSchema(schema.key);
```

```
    assert.deepStrictEqual(foundSchema, schema);
  });
  it('should find key in schema provided as parameter', async () => {
    const schema: ERC725JSONSchema = {
      name: 'ParameterSchema',
      key: '0x777f55baf2e0c9f73d3bb456dfb8dbf6e609bf557969e3184c17ff925b3c402c',
      keyType: 'Singleton',
      valueContent: 'JSONURL',
      valueType: 'bytes',
    };

    const erc725 = new ERC725([]);

    const foundSchema = erc725.getSchema(schema.key, [schema]);

    assert.deepStrictEqual(foundSchema, schema);
  });
});

describe('encodeKeyName', () => {
  const erc725Instance = new ERC725([]);

  it('is available on instance and class', () => {
    assert.deepStrictEqual(
      ERC725.encodeKeyName('MyKeyName'),
      '0x35e6950bc8d21a1699e58328a3c4066df5803bb0b570d0150cb3819288e764b2',
    );
    assert.deepStrictEqual(
      erc725Instance.encodeKeyName('MyKeyName'),
      '0x35e6950bc8d21a1699e58328a3c4066df5803bb0b570d0150cb3819288e764b2',
    );
  });

  it('works for dynamic keys', () => {
    assert.deepStrictEqual(
      ERC725.encodeKeyName(
        'FavouriteFood:<address>',
        '0xa4FBbFe353124E6fa6Bb7f8e088a9269dF552EA2',
      ),
      '0x31145577efe228036af40000a4fbbfe353124e6fa6bb7f8e088a9269df552ea2',
    );
    assert.deepStrictEqual(
      erc725Instance.encodeKeyName(
        'FavouriteFood:<address>',
        '0xa4FBbFe353124E6fa6Bb7f8e088a9269dF552EA2',
      ),
      '0x31145577efe228036af40000a4fbbfe353124e6fa6bb7f8e088a9269df552ea2',
    );
  });
});
```

```
describe('supportsInterface', () => {
  const erc725Instance = new ERC725([]);

  it('is available on instance and class', () => {
    assert.typeOf(ERC725.supportsInterface, 'function');
    assert.typeOf(erc725Instance.supportsInterface, 'function');
  });

  const interfaceId = INTERFACE_IDS_0_12_0.LSP1UniversalReceiver;
  const rpcUrl = 'https://my.test.provider';
  const contractAddress = '0xcafecafecafecafecafecafecafecafecafecafe';

  it('should throw when provided address is not an address', async () => {
    try {
      await ERC725.supportsInterface(interfaceId, {
        address: 'notAnAddress',
        rpcUrl,
      });
    } catch (error: any) {
      assert.deepStrictEqual(error.message, 'Invalid address');
    }
  });

  it('should throw when rpcUrl is not provided on non instantiated class', async () => {
    try {
      await ERC725.supportsInterface(interfaceId, {
        address: contractAddress,
        // @ts-ignore
        rpcUrl: undefined,
      });
    } catch (error: any) {
      assert.deepStrictEqual(error.message, 'Missing RPC URL');
    }
  });

  // TODO: add test to test the actual behavior of the function.
});

describe('checkPermissions', () => {
  const erc725Instance = new ERC725([]);

  it('is available on instance', () => {
    assert.typeOf(erc725Instance.checkPermissions, 'function');

    const requiredPermissions = [
      '0x0000000000000000000000000000000000000000000000000000000000000004',
      '0x0000000000000000000000000000000000000000000000000000000000000800',
    ];
    const grantedPermissions =
      '0x000000000000000000000000000000000000000000000000000000000000ff51';
    const result = erc725Instance.checkPermissions(
      requiredPermissions,
```

```
    grantedPermissions,
  );

  assert.equal(result, false);
});

it('is available on class', () => {
  assert.typeOf(ERC725.checkPermissions, 'function');

  const requiredPermissions = [
    '0x0000000000000000000000000000000000000000000000000000000000000004',
    '0x0000000000000000000000000000000000000000000000000000000000000800',
  ];
  const grantedPermissions =
    '0x000000000000000000000000000000000000000000000000000000000000ff51';

  const result = ERC725.checkPermissions(
    requiredPermissions,
    grantedPermissions,
  );

  assert.equal(result, false);
});
});

describe('decodeMappingKey', () => {
const erc725Instance = new ERC725([]);

it('is available on instance and class', () => {
  assert.deepStrictEqual(
    ERC725.decodeMappingKey(
      '0x35e6950bc8d21a1699e50000cafecafecafecafecafecafecafecafecafecafe',
      'MyKeyName:<address>',
    ),
    [
      {
        type: 'address',
        value: '0xCAfEcAfeCAfECaFeCaFecaFecaFECafECafeCaFe',
      },
    ],
  );
  assert.deepStrictEqual(
    erc725Instance.decodeMappingKey(
      '0x35e6950bc8d21a1699e50000cafecafecafecafecafecafecafecafecafecafe',
      'MyKeyName:<address>',
    ),
    [
      {
        type: 'address',
        value: '0xCAfEcAfeCAfECaFeCaFecaFecaFECafECafeCaFe',
      },
    ],
```

```
    );
  });
});
```

</file>

<file>

path: /src/index.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/index.ts

```
/*
    This file is part of @erc725/erc725.js.
    @erc725/erc725.js is free software: you can redistribute it and/or modify
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    @erc725/erc725.js is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.
    You should have received a copy of the GNU Lesser General Public License
    along with @erc725/erc725.js.  If not, see <http://www.gnu.org/licenses/>.
*/

/**
 * @file index.ts
 * @author Robert McLeod <@robertdavid010>
 * @author Fabian Vogelsteller <fabian@lukso.network>
 * @author Hugo Masclet <@Hugoo>
 * @date 2020
 */

import { hexToNumber, isAddress, leftPad, toHex } from 'web3-utils';
import HttpProvider from 'web3-providers-http';

import { ProviderWrapper } from './provider/providerWrapper';

import {
  encodeData,
  convertIPFSGatewayUrl,
  generateSchemasFromDynamicKeys,
} from './lib/utils';

import { getSchema } from './lib/schemaParser';
import { isValidSignature } from './lib/isValidSignature';

import {
  LSP6_ALL_PERMISSIONS,
  LSP6_DEFAULT_PERMISSIONS,
```

```
  DEFAULT_GAS_VALUE,
} from './constants/constants';
import { encodeKeyName, isDynamicKeyName } from './lib/encodeKeyName';

// Types
import { ERC725Config, ERC725Options } from './types/Config';
import { Permissions } from './types/Method';
import {
  ERC725JSONSchema,
  ERC725JSONSchemaKeyType,
  ERC725JSONSchemaValueContent,
  ERC725JSONSchemaValueType,
} from './types/ERC725JSONSchema';
import {
  DecodeDataInput,
  DecodeDataOutput,
  EncodeDataInput,
  FetchDataOutput,
} from './types/decodeData';
import { GetDataDynamicKey, GetDataInput } from './types/GetData';
import { decodeData } from './lib/decodeData';
import { getDataFromExternalSources } from './lib/getDataFromExternalSources';
import { DynamicKeyPart, DynamicKeyParts } from './types/dynamicKeys';
import { getData } from './lib/getData';
import { supportsInterface, checkPermissions } from './lib/detector';
import { decodeMappingKey } from './lib/decodeMappingKey';

export {
  ERC725JSONSchema,
  ERC725JSONSchemaKeyType,
  ERC725JSONSchemaValueContent,
  ERC725JSONSchemaValueType,
};

export { ERC725Config, KeyValuePair, ProviderTypes } from './types';
export { encodeData } from './lib/utils';

/**
 * This package is currently in early stages of development, <br/>use only for testing or experimentation purposes.<br/>
 *
 * @typeParam Schema
 *
 */
export class ERC725 {
  options: ERC725Options;

  /**
   * Creates an instance of ERC725.
   * @param {ERC725JSONSchema[]} schema More information available here: [LSP-2-ERC725YJSONSchema](https://github.com/lukso-network/LIPs/blob/master/LSPs/LSP-2-ERC725YJSONSchema.md)
```

```
   * @param {string} address Address of the ERC725 contract you want to interact with
   * @param {any} provider
   * @param {ERC725Config} config Configuration object.
   *
   */
  constructor(
    schemas: ERC725JSONSchema[],
    address?,
    provider?: any,
    config?: ERC725Config,
  ) {
    // NOTE: provider param can be either the provider, or and object with {provider:xxx ,type:xxx}

    // TODO: Add check for schema format?
    if (!schemas) {
      throw new Error('Missing schema.');
    }

    const defaultConfig = {
      ipfsGateway: 'https://cloudflare-ipfs.com/ipfs/',
      gas: DEFAULT_GAS_VALUE,
    };

    this.options = {
      schemas: this.validateSchemas(schemas),
      address,
      provider: ERC725.initializeProvider(
        provider,
        config?.gas ? config?.gas : defaultConfig.gas,
      ),
      ipfsGateway: config?.ipfsGateway
        ? convertIPFSGatewayUrl(config?.ipfsGateway)
        : defaultConfig.ipfsGateway,
      gas: config?.gas ? config?.gas : defaultConfig.gas,
    };
  }

  /**
   * To prevent weird behovior from the lib, we must make sure all the schemas are correct before loading them.
   *
   * @param schemas
   * @returns
   */
  // eslint-disable-next-line class-methods-use-this
  private validateSchemas(schemas: ERC725JSONSchema[]) {
    return schemas.filter((schema) => {
      try {
        const encodedKeyName = encodeKeyName(schema.name);

        const isKeyValid = schema.key === encodedKeyName;
```

```
      if (!isKeyValid) {
        console.log(
           The schema with keyName: ${schema.key} is skipped because its key hash does not match its key name (expected: ${encodedKeyName}, got: ${schema.key}). ,
        );
      }

      return isKeyValid;
    } catch (err: any) {
      // We could not encodeKeyName, probably because the key is dynamic (Mapping or MappingWithGrouping).

      // TODO: make sure the dynamic key name is valid:
      // - has max 2 variables
      // - variables are correct (<string>, <bool>, etc.)

      // Keeping dynamic keys may be an issue for getData / fetchData functions.
      return true;
    }
  });
}

private static initializeProvider(providerOrRpcUrl, gasInfo) {
  // do not fail on no-provider
  if (!providerOrRpcUrl) return undefined;

  // if provider is a string, assume it's a rpcUrl
  if (typeof providerOrRpcUrl === 'string') {
    return new ProviderWrapper(new HttpProvider(providerOrRpcUrl), gasInfo);
  }

  if (
    typeof providerOrRpcUrl.request === 'function' ||
    typeof providerOrRpcUrl.send === 'function'
  )
    return new ProviderWrapper(providerOrRpcUrl, gasInfo);

  throw new Error( Incorrect or unsupported provider ${providerOrRpcUrl} );
}
/**
 * Gets **decoded data** for one, many or all keys of the specified  ERC725  smart-contract.
 * When omitting the  keyOrKeys  parameter, it will get all the keys (as per {@link ERC725JSONSchema | ERC725JSONSchema} definition).
 *
 * Data returned by this function does not contain external data of [ JSONURL ](https://github.com/lukso-network/LIPs/blob/master/LSPs/LSP-2-ERC725YJSONSchema.md#jsonurl)
 * or [ ASSETURL ](https://github.com/lukso-network/LIPs/blob/master/LSPs/LSP-2-ERC725YJSONSchema.md#asseturl) schema elements.
 *
 * If you would like to receive everything in one go, you can use {@link ERC725.fetchData |  fetchData } for that.
 *
```

```
   * @param {*} keyOrKeys The name (or the encoded name as the schema 'key') of the schema element in the class instance's schema.
   *
   * @returns If the input is an array: an object with schema element key names as properties, with corresponding **decoded** data as values. If the input is a string, it directly returns the **decoded** data.
   */
  async getData(
    keyOrKeys?: Array<string | GetDataDynamicKey>,
  ): Promise<DecodeDataOutput[]>;
  async getData(
    keyOrKeys?: string | GetDataDynamicKey,
  ): Promise<DecodeDataOutput>;
  async getData(
    keyOrKeys?: GetDataInput,
  ): Promise<DecodeDataOutput | DecodeDataOutput[]> {
    this.getAddressAndProvider();
    return getData(this.options, keyOrKeys);
  }

  /**
   * Since {@link ERC725.getData |  getData } exclusively returns data that is stored on the blockchain,  fetchData  comes in handy.
   * Additionally to the data from the blockchain,  fetchData  also returns data from IPFS or HTTP(s) endpoints
   * stored as [ JSONURL ](https://github.com/lukso-network/LIPs/blob/master/LSPs/LSP-2-ERC725YJSONSchema.md#jsonurl) or [ ASSETURL ](https://github.com/lukso-network/LIPs/blob/master/LSPs/LSP-2-ERC725YJSONSchema.md#asseturl).
   *
   * To ensure **data authenticity**  fetchData  compares the  hash  of the fetched JSON with the  hash  stored on the blockchain.
   *
   * @param {*} keyOrKeys The name (or the encoded name as the schema 'key') of the schema element in the class instance's schema.
   *
   * @returns Returns the fetched and decoded value depending 'valueContent' for the schema element, otherwise works like getData.
   */

  async fetchData(
    keyOrKeys?: Array<string | GetDataDynamicKey>,
  ): Promise<FetchDataOutput[]>;
  async fetchData(
    keyOrKeys?: string | GetDataDynamicKey,
  ): Promise<FetchDataOutput>;
  async fetchData(
    keyOrKeys?: GetDataInput,
  ): Promise<FetchDataOutput | FetchDataOutput[]> {
    let keyNames: Array<string | GetDataDynamicKey>;

    if (Array.isArray(keyOrKeys)) {
      keyNames = keyOrKeys;
    } else if (!keyOrKeys) {
```

```
      keyNames = this.options.schemas
        .map((element) => element.name)
        .filter((key) => !isDynamicKeyName(key));
    } else {
      keyNames = [keyOrKeys];
    }

    const dataFromChain = await this.getData(keyNames);

    // NOTE: this step is executed in getData function above
    // We can optimize by computing it only once.
    const schemas = generateSchemasFromDynamicKeys(
      keyNames,
      this.options.schemas,
    );

    const dataFromExternalSources = await getDataFromExternalSources(
      schemas,
      dataFromChain,
      this.options.ipfsGateway,
    );

    if (
      keyOrKeys &&
      !Array.isArray(keyOrKeys) &&
      dataFromExternalSources.length > 0
    ) {
      return dataFromExternalSources[0];
    }

    return dataFromExternalSources;
  }

  /**
   * Parses a hashed key or a list of hashed keys and will attempt to return its corresponding LSP-2 ERC725YJSONSchema object.
   * The function will look for a corresponding key within the schemas:
   *  - in  ./schemas  folder
   *  - provided at initialisation
   *  - provided in the function call
   *
   * @param keyOrKeys The hashed key or array of keys for which you want to find the corresponding LSP-2 ERC725YJSONSchema.
   * @param providedSchemas If you provide your own ERC725JSONSchemas, the parser will also try to find a key match against these schemas.
   */
  getSchema(
    keyOrKeys: string[],
    providedSchemas?: ERC725JSONSchema[],
  ): Record<string, ERC725JSONSchema | null>;
  getSchema(
    keyOrKeys: string,
```

```
    providedSchemas?: ERC725JSONSchema[],
  ): ERC725JSONSchema | null;
  getSchema(
    keyOrKeys: string | string[],
    providedSchemas?: ERC725JSONSchema[],
  ): ERC725JSONSchema | null | Record<string, ERC725JSONSchema | null> {
    return getSchema(
      keyOrKeys,
      this.options.schemas.concat(providedSchemas || []),
    );
  }

  /**
   * To be able to store your data on the blockchain, you need to encode it according to your {@link
ERC725JSONSchema}.
   *
   * @param {{ [key: string]: any }} data An object with one or many properties, containing the data that
needs to be encoded.
   * @param schemas Additionnal ERC725JSONSchemas which will be concatenated with the schemas
provided on init.
   *
   * @returns An object with hashed keys and encoded values.
   *
   * When encoding JSON it is possible to pass in the JSON object and the URL where it is available
publicly.
   * The JSON will be hashed with &amp;nbsp;keccak256&amp;nbsp;.
   */
  encodeData(data: EncodeDataInput[], schemas?: ERC725JSONSchema[]) {
    return encodeData(
      data,
      Array.prototype.concat(this.options.schemas, schemas),
    );
  }

  /**
   * To be able to store your data on the blockchain, you need to encode it according to your {@link
ERC725JSONSchema}.
   *
   * @param {{ [key: string]: any }} data An object with one or many properties, containing the data that
needs to be encoded.
   * @param schemas ERC725JSONSchemas which will be used to encode the keys.
   *
   * @returns An object with hashed keys and encoded values.
   *
   * When encoding JSON it is possible to pass in the JSON object and the URL where it is available
publicly.
   * The JSON will be hashed with &amp;nbsp;keccak256&amp;nbsp;.
   */
  static encodeData(data: EncodeDataInput[], schemas: ERC725JSONSchema[]) {
    return encodeData(data, schemas);
  }
```

```
 /**
  * In case you are reading the key-value store from an ERC725 smart-contract key-value store
  * without  @erc725/erc725.js  you can use  decodeData  to do the decoding for
you.
  *
  * It is more convenient to use {@link ERC725.fetchData |  fetchData }.
  * It does the  decoding  and  fetching  of external references for you
automatically.
  *
  * @param {{ [key: string]: any }} data An object with one or many properties.
  * @param schemas ERC725JSONSchemas which will be used to encode the keys.
  *
  * @returns Returns decoded data as defined and expected in the schema:
  */
 decodeData(
   data: DecodeDataInput[],
   schemas?: ERC725JSONSchema[],
 ): { [key: string]: any } {
   return decodeData(
     data,
     Array.prototype.concat(this.options.schemas, schemas),
   );
 }

 /**
  * In case you are reading the key-value store from an ERC725 smart-contract key-value store
  * without  @erc725/erc725.js  you can use  decodeData  to do the decoding for
you.
  *
  * It is more convenient to use {@link ERC725.fetchData |  fetchData }.
  * It does the  decoding  and  fetching  of external references for you
automatically.
  *
  * @param {{ [key: string]: any }} data An object with one or many properties.
  * @param schemas ERC725JSONSchemas which will be used to encode the keys.
  *
  * @returns Returns decoded data as defined and expected in the schema:
  */
 static decodeData(
   data: DecodeDataInput[],
   schemas: ERC725JSONSchema[],
 ): { [key: string]: any } {
   return decodeData(data, schemas);
 }

 /**
  * An added utility method which simply returns the owner of the contract.
  * Not directly related to ERC725 specifications.
  *
  * @param {string} [address]
  * @returns The address of the contract owner as stored in the contract.
  *
```

```
   *    This method is not yet supported when using the  graph  provider type.
   *
   *    javascript title="Example"
   * await myERC725.getOwner();
   * // '0x94933413384997F9402cc07a650e8A34d60F437A'
   *
   * await myERC725.getOwner("0x3000783905Cc7170cCCe49a4112Deda952DDBe24");
   * // '0x7f1b797b2Ba023Da2482654b50724e92EB5a7091'
   *    
   */
  async getOwner(_address?: string) {
    const { address, provider } = this.getAddressAndProvider();

    return provider.getOwner(_address || address);
  }

  /**
   * A helper function which checks if a signature is valid according to the EIP-1271 standard.
   *
   * @param messageOrHash if it is a 66 chars string with 0x prefix, it will be considered as a hash (keccak256). If not, the message will be wrapped as follows: "\x19Ethereum Signed Message:\n" + message.length + message and hashed.
   * @param signature
   * @returns true if isValidSignature call on the contract returns the magic value. false otherwise
   */
  async isValidSignature(
    messageOrHash: string,
    signature: string,
  ): Promise<boolean> {
    if (!this.options.address || !isAddress(this.options.address)) {
      throw new Error('Missing ERC725 contract address.');
    }
    if (!this.options.provider) {
      throw new Error('Missing provider.');
    }

    return isValidSignature(
      messageOrHash,
      signature,
      this.options.address,
      this.options.provider,
    );
  }

  private getAddressAndProvider() {
    if (!this.options.address || !isAddress(this.options.address)) {
      throw new Error('Missing ERC725 contract address.');
    }
    if (!this.options.provider) {
      throw new Error('Missing provider.');
    }
```

```
    return {
      address: this.options.address,
      provider: this.options.provider,
    };
  }

  /**
   * Encode permissions into a hexadecimal string as defined by the LSP6 KeyManager Standard.
   *
   * @link https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-6-KeyManager.md LSP6 KeyManager Standard.
   * @param permissions The permissions you want to specify to be included or excluded. Any ommitted permissions will default to false.
   * @returns {*} The permissions encoded as a hexadecimal string as defined by the LSP6 Standard.
   */
  static encodePermissions(permissions: Permissions): string {
    const result = Object.keys(permissions).reduce((previous, key) => {
      return permissions[key]
        ? previous | Number(hexToNumber(LSP6_DEFAULT_PERMISSIONS[key]))
        : previous;
    }, 0);

    return leftPad(toHex(result), 64);
  }

  /**
   * Encode permissions into a hexadecimal string as defined by the LSP6 KeyManager Standard.
   *
   * @link https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-6-KeyManager.md LSP6 KeyManager Standard.
   * @param permissions The permissions you want to specify to be included or excluded. Any ommitted permissions will default to false.
   * @returns {*} The permissions encoded as a hexadecimal string as defined by the LSP6 Standard.
   */
  encodePermissions(permissions: Permissions): string {
    return ERC725.encodePermissions(permissions);
  }

  /**
   * Decodes permissions from hexadecimal as defined by the LSP6 KeyManager Standard.
   *
   * @link https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-6-KeyManager.md LSP6 KeyManager Standard.
   * @param permissionHex The permission hexadecimal value to be decoded.
   * @returns Object specifying whether default LSP6 permissions are included in provided hexademical string.
   */
  static decodePermissions(permissionHex: string) {
    const result = {
      CHANGEOWNER: false,
      ADDCONTROLLER: false,
      EDITPERMISSIONS: false,
```

```
    ADDEXTENSIONS: false,
    CHANGEEXTENSIONS: false,
    ADDUNIVERSALRECEIVERDELEGATE: false,
    CHANGEUNIVERSALRECEIVERDELEGATE: false,
    REENTRANCY: false,
    SUPER_TRANSFERVALUE: false,
    TRANSFERVALUE: false,
    SUPER_CALL: false,
    CALL: false,
    SUPER_STATICCALL: false,
    STATICCALL: false,
    SUPER_DELEGATECALL: false,
    DELEGATECALL: false,
    DEPLOY: false,
    SUPER_SETDATA: false,
    SETDATA: false,
    ENCRYPT: false,
    DECRYPT: false,
    SIGN: false,
    EXECUTE_RELAY_CALL: false,
  };

  const permissionsToTest = Object.keys(LSP6_DEFAULT_PERMISSIONS);
  if (permissionHex === LSP6_ALL_PERMISSIONS) {
    permissionsToTest.forEach((testPermission) => {
      result[testPermission] = true;
    });
    return result;
  }

  const passedPermissionDecimal = Number(hexToNumber(permissionHex));

  permissionsToTest.forEach((testPermission) => {
    const decimalTestPermission = Number(
      hexToNumber(LSP6_DEFAULT_PERMISSIONS[testPermission]),
    );
    const isPermissionIncluded =
      (passedPermissionDecimal & decimalTestPermission) ===
      decimalTestPermission;

    result[testPermission] = isPermissionIncluded;
  });

  return result;
}

/**
 * Decodes permissions from hexadecimal as defined by the LSP6 KeyManager Standard.
 *
 * @link https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-6-KeyManager.md LSP6 KeyManager Standard.
 * @param permissionHex The permission hexadecimal value to be decoded.
```

```
   * @returns Object specifying whether default LSP6 permissions are included in provided hexademical string.
   */
  decodePermissions(permissionHex: string) {
    return ERC725.decodePermissions(permissionHex);
  }

  /**
   * Hashes a key name for use on an ERC725Y contract according to LSP2 ERC725Y JSONSchema standard.
   *
   * @param {string} keyName The key name you want to encode.
   * @param {DynamicKeyParts} dynamicKeyParts String or Array of String values used to construct the key.
   * @link https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-2-ERC725YJSONSchema.md ERC725YJsonSchema standard.
   * @returns {string} The keccak256 hash of the provided key name. This is the key that must be retrievable from the ERC725Y contract via ERC725Y.getData(bytes32 key).
   */
  static encodeKeyName(
    keyName: string,
    dynamicKeyParts?: DynamicKeyParts,
  ): string {
    return encodeKeyName(keyName, dynamicKeyParts);
  }

  /**
   * Hashes a key name for use on an ERC725Y contract according to LSP2 ERC725Y JSONSchema standard.
   *
   * @param {string} keyName The key name you want to encode.
   * @param {DynamicKeyParts} dynamicKeyParts String or Array of String values used to construct the key.
   * @link https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-2-ERC725YJSONSchema.md ERC725YJsonSchema standard.
   * @returns {string} The keccak256 hash of the provided key name. This is the key that must be retrievable from the ERC725Y contract via ERC725Y.getData(bytes32 key).
   */
  encodeKeyName(keyName: string, dynamicKeyParts?: DynamicKeyParts): string {
    return encodeKeyName(keyName, dynamicKeyParts);
  }

  /**
   * Decodes a hashed key used on an ERC725Y contract according to LSP2 ERC725Y JSONSchema standard.
   *
   * @param {string} keyHash Key hash that needs to be decoded.
   * @param {string | ERC725JSONSchema} keyNameOrSchema Key name following schema specifications or ERC725Y JSON Schema to follow in order to decode the key.
   * @link https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-2-ERC725YJSONSchema.md ERC725YJsonSchema standard.
   * @returns {DynamicKeyPart[]} The Array with all the key decoded dynamic parameters. Each object have an attribute type and value.
   */
```

```
  static decodeMappingKey(
    keyHash: string,
    keyNameOrSchema: string | ERC725JSONSchema,
  ): DynamicKeyPart[] {
    return decodeMappingKey(keyHash, keyNameOrSchema);
  }

  /**
   * Decodes a hashed key used on an ERC725Y contract according to LSP2 ERC725Y JSONSchema standard.
   *
   * @param {string} keyHash Key hash that needs to be decoded.
   * @param {string | ERC725JSONSchema} keyNameOrSchema Key name following schema specifications or ERC725Y JSON Schema to follow in order to decode the key.
   * @link https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-2-ERC725YJSONSchema.md ERC725YJsonSchema standard.
   * @returns {DynamicKeyPart[]} The Array with all the key decoded dynamic parameters. Each object have an attribute type and value.
   */
  decodeMappingKey(
    keyHash: string,
    keyNameOrSchema: string | ERC725JSONSchema,
  ): DynamicKeyPart[] {
    return decodeMappingKey(keyHash, keyNameOrSchema);
  }

  /**
   * Check if the ERC725 object supports
   * a certain interface.
   *
   * @param interfaceIdOrName Interface ID or supported interface name.
   * @returns {Promise<boolean>} if interface is supported.
   */
  async supportsInterface(interfaceIdOrName: string): Promise<boolean> {
    const { address, provider } = this.getAddressAndProvider();

    return supportsInterface(interfaceIdOrName, {
      address,
      provider,
    });
  }

  /**
   * Check if a smart contract address
   * supports a certain interface.
   *
   * @param {string} interfaceIdOrName Interface ID or supported interface name.
   * @param options Object of address, RPC URL and optional gas.
   * @returns {Promise<boolean>} if interface is supported.
   */
  static async supportsInterface(
    interfaceIdOrName: string,
```

```
    options: { address: string; rpcUrl: string; gas?: number },
  ): Promise<boolean> {
    if (!isAddress(options.address)) {
      throw new Error('Invalid address');
    }
    if (!options.rpcUrl) {
      throw new Error('Missing RPC URL');
    }

    return supportsInterface(interfaceIdOrName, {
      address: options.address,
      provider: this.initializeProvider(
        options.rpcUrl,
        options?.gas ? options?.gas : DEFAULT_GAS_VALUE,
      ),
    });
  }

  /**
   * Check if the required permissions are included in the granted permissions as defined by the LSP6 KeyManager Standard.
   *
   * @link https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-6-KeyManager.md LSP6 KeyManager Standard.
   * @param requiredPermissions An array of required permissions or a single required permission.
   * @param grantedPermissions The granted permissions as a 32-byte hex string.
   * @return A boolean value indicating whether the required permissions are included in the granted permissions.
   */
  static checkPermissions(
    requiredPermissions: string[] | string,
    grantedPermissions: string,
  ): boolean {
    return checkPermissions(requiredPermissions, grantedPermissions);
  }

  /**
   * Check if the required permissions are included in the granted permissions as defined by the LSP6 KeyManager Standard.
   *
   * @link https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-6-KeyManager.md LSP6 KeyManager Standard.
   * @param requiredPermissions An array of required permissions or a single required permission.
   * @param grantedPermissions The granted permissions as a 32-byte hex string.
   * @return A boolean value indicating whether the required permissions are included in the granted permissions.
   */
  checkPermissions(
    requiredPermissions: string[] | string,
    grantedPermissions: string,
  ): boolean {
    return ERC725.checkPermissions(requiredPermissions, grantedPermissions);
```

```
  }
}

export default ERC725;
```

</file>

<file>

path: /src/lib/decodeData.test.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/lib/decodeData.test.ts

```
/*
    This file is part of @erc725/erc725.js.
    @erc725/erc725.js is free software: you can redistribute it and/or modify
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    @erc725/erc725.js is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.
    You should have received a copy of the GNU Lesser General Public License
    along with web3.js.  If not, see <http://www.gnu.org/licenses/>.
*/

/* eslint-disable no-unused-expressions */

import { expect } from 'chai';

import { ERC725JSONSchema } from '../types/ERC725JSONSchema';
import { decodeData, decodeTupleKeyValue, isValidTuple } from './decodeData';

describe('decodeData', () => {
  const schemas: ERC725JSONSchema[] = [
    {
      name: 'KeyOne',
      key: '0x0f4e1c48ef0a593fadd83075e26a9418949fca5ebd47f3224508df4dab7584d9',
      keyType: 'Singleton',
      valueType: 'bytes',
      valueContent: '0x1111',
    },
    {
      name: 'KeyTwo',
      key: '0x13e32c6169bcd9107cb714c65a5cc4dbd09d437bee789d0c735467d3af8dc3b0',
      keyType: 'Singleton',
      valueType: 'bytes',
      valueContent: '0x2222',
    },
    {
```

```
    name: 'MyKeyName:<bytes32>:<bool>',
    key: '0x',
    keyType: 'Singleton',
    valueType: 'bytes',
    valueContent: 'JSONURL',
  },
  {
    name: 'MyDynamicKey:<address>',
    key: '0x',
    keyType: 'Singleton',
    valueType: 'bytes',
    valueContent: 'JSONURL',
  },
];

it('decodes each key', () => {
  const decodedData = decodeData(
    [
      {
        keyName: 'KeyOne',
        value: '0x1111',
      },
      {
        keyName: 'KeyTwo',
        value: '0x2222',
      },
    ],
    schemas,
  );

  expect(decodedData.map(({ name }) => name)).to.eql(['KeyOne', 'KeyTwo']);
});

it('parses non array input correctly', () => {
  const decodedData = decodeData(
    {
      keyName: 'KeyOne',
      value: '0x1111',
    },
    schemas,
  );

  expect(decodedData.name).to.eql('KeyOne');
});

it('parses type tuples/Mixed correctly', () => {
  const schema: ERC725JSONSchema = {
    name: 'MyDynamicKey:<address>',
    key: '0x',
    keyType: 'Singleton',
    valueType: '(bytes4,bytes8)',
    valueContent: '(Bytes4,Number)',
```

```
  };

  const decodedData = decodeData(
    {
      keyName: 'MyDynamicKey:<address>',
      dynamicKeyParts: '0xcafecafecafecafecafecafecafecafecafecafe',
      value: '0x11223344000000000000000c',
      //        |------||--------------|
      //         bytes4     bytes8
      //         bytes4     number
    },
    [schema],
  );

  expect(decodedData.value).to.eql(['0x11223344', 12]);
});

it('parses type Array correctly', () => {
  const decodedData = decodeData(
    {
      keyName: 'LSP12IssuedAssets[]',
      value: [
        {
          key: '0x7c8c3416d6cda87cd42c71ea1843df28ac4850354f988d55ee2eaa47b6dc05cd',
          value: '0x0000000000000000000000000000002',
        },
        {
          key: '0x7c8c3416d6cda87cd42c71ea1843df2800000000000000000000000000000000',
          value: '0xd943553d9b005b3c0a9da169b768a31c57844e490',
        },
        {
          key: '0x7c8c3416d6cda87cd42c71ea1843df2800000000000000000000000000000001',
          value: '0xdaea594e385fc724449e3118b2db7e86dfba1826',
        },
      ],
    },
    [
      {
        name: 'LSP12IssuedAssets[]',
        key: '0x7c8c3416d6cda87cd42c71ea1843df28ac4850354f988d55ee2eaa47b6dc05cd',
        keyType: 'Array',
        valueContent: 'Address',
        valueType: 'address',
      },
    ],
  );

  expect(decodedData.name).to.eql('LSP12IssuedAssets[]');
  expect(decodedData.value).to.eql([
    '0xD943553D9B005B3c0A9Da169b768a31C57844e490',
    '0xDaea594E385Fc724449E3118B2Db7E86dFBa1826',
  ]);
```

```
  });

  it('decodes dynamic keys', () => {
    const decodedData = decodeData(
      [
        {
          keyName: 'MyKeyName:<bytes32>:<bool>',
          dynamicKeyParts: [
            '0xaaaabbbbccccddddeeeeffff111122223333444455556666777788889999aaaa',
            'true',
          ],
          value:

'0x6f357c6a820464ddfac1bec070cc14a8daf04129871d458f2ca94368aae8391311af6361697066733a2f2f51
6d597231564a4c776572673670456f73636468564775676f3339706136727963455a4c6a745250446657383
4554178',
        },
        {
          keyName: 'MyDynamicKey:<address>',
          dynamicKeyParts: '0xcafecafecafecafecafecafecafecafecafecafe',
          value:

'0x6f357c6a820464ddfac1bec070cc14a8daf04129871d458f2ca94368aae8391311af6361697066733a2f2f51
6d597231564a4c776572673670456f73636468564775676f3339706136727963455a4c6a745250446657383
4554178',
        },
        {
          keyName: 'KeyTwo',
          value: '0x2222',
        },
      ],
      schemas,
    );

    expect(decodedData.map(({ name }) => name)).to.eql([
      'MyKeyName:aaaabbbbccccddddeeeeffff111122223333444455556666777788889999aaaa:true',
      'MyDynamicKey:cafecafecafecafecafecafecafecafecafecafe',
      'KeyTwo',
    ]);
  });
});

describe('tuple', () => {
  describe('decodeTupleKeyValue', () => {
    const testCases = [
      {
        valueContent: '(Bytes4,Number)',
        valueType: '(bytes4,bytes8)',
        encodedValue: '0xdeadbeaf000000000000000c',
        decodedValue: ['0xdeadbeaf', 12],
      },
    ]; // TODO: add more cases? Address, etc.
```

```
  testCases.forEach((testCase) => {
    it(&amp;nbsp;decodes tuple values&amp;nbsp;;, () => {
      expect(
        decodeTupleKeyValue(
          testCase.valueContent,
          testCase.valueType,
          testCase.encodedValue,
        ),
      ).to.eql(testCase.decodedValue);
    });
  });
});

describe('isValidTupleValueType', () => {
  const testCases = [
    {
      valueType: 'abcd', // not a tuple
      valueContent: 'WebThr33',
      isTuple: false,
    },
    {
      valueType: '()',
      valueContent: '()',
      isTuple: false, // it is empty
    },
    {
      valueType: '(bytes4)',
      valueContent: '(HeyHey)',
      isTuple: false, // valueContent is wrong
      shouldThrow: true,
    },
    {
      valueType: '(bytes4,number)', // number not allowed in valueType
      valueContent: '(Bytes4,Number)',
      isTuple: false,
      shouldThrow: true,
    },
    {
      valueType: '(bytes4,bytes8)',
      valueContent: '(Bytes4,Number,Bytes5)',
      isTuple: false, // valueContent length != valueType length
      shouldThrow: true,
    },
    {
      valueType: '(bytes4,bytes8)',
      valueContent: '(Bytes2,Number)',
      isTuple: false, // first item in valueType does not fit inside first item in valueContent (bytes4 > bytes2)
      shouldThrow: true,
    },
    {
      valueType: '(bytes4,bytes8)',
```

```
      valueContent: '(Bytes8,Number)',
      isTuple: true, // first item in valueType fit in first item of valueContent (bytes4 < bytes8)
    },
    {
      valueType: '(bytes4,bytes8)',
      valueContent: '(Bytes4,Number)',
      isTuple: true,
    },
    {
      valueType: '(bytes4,bytes8,bytes16)',
      valueContent: '(Bytes4,Number,Bytes16)',
      isTuple: true,
    },
    {
      valueType: '(bytes4,bytes4)',
      valueContent: '(Number,0x112233XX)',
      isTuple: false,
      shouldThrow: true, // valueContent is not a valid hex value
    },
    {
      valueType: '(bytes4,bytes4)',
      valueContent: '(Number,0x1122334455)',
      isTuple: false,
      shouldThrow: true, // valueContent is bytes5 vs bytes4
    },
  ];

  testCases.forEach((testCase) => {
    it( detects valueType: ${testCase.valueType} valueContent: ${
      testCase.valueContent
    } as ${testCase.isTuple ? 'tuple' : 'non tuple'} ;, () => {
      if (testCase.shouldThrow) {
        expect(() => {
          isValidTuple(testCase.valueType, testCase.valueContent);
        }).to.throw();
        return;
      }

      expect(
        isValidTuple(testCase.valueType, testCase.valueContent),
      ).to.equal(testCase.isTuple);
    });
  });
 });
});
```

</file>

<file>

path: /src/lib/decodeData.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/lib/decodeData.ts

```
/*
    This file is part of @erc725/erc725.js.
    @erc725/erc725.js is free software: you can redistribute it and/or modify
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    @erc725/erc725.js is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.
    You should have received a copy of the GNU Lesser General Public License
    along with @erc725/erc725.js.  If not, see <http://www.gnu.org/licenses/>.
*/

/**
 * @file lib/decodeData.ts
 * @author Robert McLeod <@robertdavid010>
 * @author Hugo Masclet <@Hugoo>
 * @author Callum Grindle <@CallumGrindle>
 * @date 2023
 */

import { isHexStrict } from 'web3-utils';
import { COMPACT_BYTES_ARRAY_STRING } from '../constants/constants';

import { DecodeDataInput, DecodeDataOutput } from '../types/decodeData';
import { ERC725JSONSchema } from '../types/ERC725JSONSchema';
import { isDynamicKeyName } from './encodeKeyName';
import { valueContentEncodingMap, decodeValueType } from './encoder';
import { getSchemaElement } from './getSchemaElement';
import { decodeKeyValue, encodeArrayKey } from './utils';

const tupleValueTypesRegex = /bytes(\d+)/;
const valueContentsBytesRegex = /Bytes(\d+)/;

export const isValidTuple = (valueType: string, valueContent: string) => {
  if (valueType.length <= 2 && valueContent.length <= 2) {
    return false;
  }

  if (
    valueType[0] !== '(' &&
    valueType[valueType.length - 1] !== ')' &&
    valueContent[0] !== '(' &&
    valueContent[valueContent.length - 1] !== ')'
  ) {
    return false;
  }

  // At this stage, we can assume the user is trying to use a tuple, let's throw errors instead of returning
```

```
  // false

  let valueTypeToDecode = valueType;

  if (valueType.includes(COMPACT_BYTES_ARRAY_STRING)) {
    valueTypeToDecode = valueType.replace(COMPACT_BYTES_ARRAY_STRING, '');
  }

  const valueTypeParts = valueTypeToDecode
    .substring(1, valueTypeToDecode.length - 1)
    .split(',');

  const valueContentParts = valueContent
    .substring(1, valueContent.length - 1)
    .split(',');

  const tuplesValidValueTypes = [
    'bytes2',
    'bytes4',
    'bytes8',
    'bytes16',
    'bytes32',
    'address',
  ];

  if (valueTypeParts.length !== valueContentParts.length) {
    throw new Error(
       Invalid tuple for valueType: ${valueType} / valueContent: ${valueContent}. They should have the
same number of elements. Got: ${valueTypeParts.length} and ${valueContentParts.length} ,
    );
  }

  for (let i = 0; i < valueTypeParts.length; i++) {
    if (!tuplesValidValueTypes.includes(valueTypeParts[i])) {
      throw new Error(
         Invalid tuple for valueType: ${valueType} / valueContent: ${valueContent}. Type:
${valueTypeParts[i]} is not valid. Valid types are: ${tuplesValidValueTypes} ,
      );
    }

    const valueTypeBytesLength = valueTypeParts[i].split('bytes')[1];

    if (
      valueTypeParts[i].match(tupleValueTypesRegex) &&
      valueContentParts[i].match(valueContentsBytesRegex)
    ) {
      const valueContentBytesLength = valueContentParts[i].slice(5);

      if (valueTypeBytesLength > valueContentBytesLength) {
        throw new Error(
           Invalid tuple (${valueType},${valueContent}: ${valueType[i]} cannot fit in
${valueContent[i]} ,
```

```
        );
      }
    }

    if (
      valueContentEncodingMap(valueContentParts[i]).type === 'unknown' &&
      valueContentParts[i].slice(0, 5) !== 'Bytes' &&
      valueContentParts[i].slice(0, 2) !== '0x'
    ) {
      throw new Error(
         Invalid tuple for valueType: ${valueType} / valueContent: ${valueContent}. valueContent of type:
${valueContentParts[i]} is not valid ,
      );
    }

    if (isHexStrict(valueContentParts[i])) {
      // check if length of a hex literal in valueContent (e.g: 0x122334455)
      // is compatible with the valueType (e.g: bytes4)
      const hexLiteralLength = valueContentParts[i].length - 2;

      if (parseInt(valueTypeBytesLength, 10) < hexLiteralLength) {
        throw new Error(
           Invalid tuple (${valueType},${valueContent}: ${valueContent[i]} cannot fit in
${valueType[i]} ,
        );
      }
    } else if (valueContentParts[i].startsWith('0x')) {
      // Value starts with 0x bit is not hex... hmmm... weird :)
      throw new Error(
         Invalid tuple for valueType: ${valueType} / valueContent: ${valueContent}. valueContent of type:
${valueContentParts[i]} is not a valid hex value ,
      );
    }
  }

  return true;
};

export const decodeTupleKeyValue = (
  valueContent: string, // i.e. (bytes4,Number,bytes16)
  valueType: string, // i.e. (bytes4,bytes8,bytes16)
  value: string, // should start with 0x
): Array<string> => {
  // We assume data has already been validated at this stage

  let valueTypeToDecode = valueType;

  if (valueType.includes('[CompactBytesArray')) {
    valueTypeToDecode = valueType.replace(COMPACT_BYTES_ARRAY_STRING, '');
  }

  const valueTypeParts = valueTypeToDecode
```

```
    .substring(1, valueTypeToDecode.length - 1)
    .split(',');
  const valueContentParts = valueContent
    .substring(1, valueContent.length - 1)
    .split(',');

  const bytesLengths: number[] = [];
  valueTypeParts.forEach((valueTypePart) => {
    const regexMatch = valueTypePart.match(tupleValueTypesRegex);

    // if we are dealing with &amp;nbsp;bytesN&amp;nbsp;
    if (regexMatch) {
      bytesLengths.push(parseInt(regexMatch[1], 10));
    }

    if (valueTypePart === 'address') bytesLengths.push(20);
  });

  const totalBytesLength = bytesLengths.reduce(
    (acc, bytesLength) => acc + bytesLength,
    0,
  );

  if (value.length !== 2 + totalBytesLength * 2) {
    console.error(
      &amp;nbsp;Trying to decode a value: ${value} which does not match the length of the valueType:
${valueType}. Expected ${totalBytesLength} bytes.&amp;nbsp;,
    );
    return [];
  }

  let cursor = 2; // to skip the 0x

  const valueParts = bytesLengths.map((bytesLength) => {
    const splitValue = value.substring(cursor, cursor + bytesLength * 2);
    cursor += bytesLength * 2;
    return &amp;nbsp;0x${splitValue}&amp;nbsp;;
  });

  return valueContentParts.map((valueContentPart, i) =>
    decodeKeyValue(valueContentPart, valueTypeParts[i], valueParts[i]),
  );
};

/**
 *
 * @param schema is an object of a schema definitions.
 * @param value will be either key-value pairs for a key type of Array, or a single value for type Singleton.
 *
 * @return the decoded value/values as per the schema definition.
 */
export function decodeKey(schema: ERC725JSONSchema, value) {
```

```
const lowerCaseKeyType = schema.keyType.toLowerCase();

switch (lowerCaseKeyType) {
  case 'array': {
    const results: any[] = [];

    // If user has requested a key which does not exist in the contract, value will be: 0x and value.find() will fail.
    if (!value || typeof value === 'string') {
      return results;
    }

    const valueElement = value.find((e) => e.key === schema.key);
    // Handle empty/non-existent array
    if (!valueElement) {
      return results;
    }

    const arrayLength =
      decodeKeyValue('Number', 'uint128', valueElement.value, schema.name) ||
      0;

    // This will not run if no match or arrayLength
    for (let index = 0; index < arrayLength; index++) {
      const dataElement = value.find(
        (e) => e.key === encodeArrayKey(schema.key, index),
      );

      if (dataElement) {
        results.push(
          decodeKeyValue(
            schema.valueContent,
            schema.valueType,
            dataElement.value,
            schema.name,
          ),
        );
      }
    } // end for loop

    return results;
  }
  case 'mappingwithgrouping':
  case 'singleton':
  case 'mapping': {
    if (Array.isArray(value)) {
      const newValue = value.find((e) => e.key === schema.key);

      // Handle empty or non-values
      if (!newValue) {
        return null;
      }
```

```
      return decodeKeyValue(
        schema.valueContent,
        schema.valueType,
        newValue.value,
        schema.name,
      );
    }

    if (schema.valueType.includes(COMPACT_BYTES_ARRAY_STRING)) {
      const valueType = schema.valueType.replace(
        COMPACT_BYTES_ARRAY_STRING,
        '',
      );
      const valueContent = schema.valueContent.replace(
        COMPACT_BYTES_ARRAY_STRING,
        '',
      );

      if (valueType[0] === '(' && valueType[valueType.length - 1] === ')') {
        const decodedCompactBytesArray = decodeValueType(
          'bytes[CompactBytesArray]',
          value,
        );
        return decodedCompactBytesArray.map((element) =>
          decodeTupleKeyValue(valueContent, valueType, element),
        );
      }
      return decodeValueType(schema.valueType, value);
    }

    if (isValidTuple(schema.valueType, schema.valueContent)) {
      return decodeTupleKeyValue(
        schema.valueContent,
        schema.valueType,
        value,
      );
    }

    return decodeKeyValue(
      schema.valueContent,
      schema.valueType,
      value,
      schema.name,
    );
  }
  default: {
    console.error(
      'Incorrect data match or keyType in schema from decodeKey(): "' +
        schema.keyType +
        '"',
    );
```

```
      return null;
    }
  }
}

/**
 * @param schema schema is an array of objects of schema definitions
 * @param data data is an array of objects of key-value pairs
 *
 * @return: all decoded data as per required by the schema and provided data
 */
export function decodeData(
  data: DecodeDataInput[],
  schema: ERC725JSONSchema[],
): DecodeDataOutput[];
export function decodeData(
  data: DecodeDataInput,
  schema: ERC725JSONSchema[],
): DecodeDataOutput;
export function decodeData(
  data: DecodeDataInput | DecodeDataInput[],
  schema: ERC725JSONSchema[],
): DecodeDataOutput | DecodeDataOutput[] {
  const processDataInput = ({
    keyName,
    dynamicKeyParts,
    value,
  }: DecodeDataInput) => {
    const isDynamic = isDynamicKeyName(keyName);

    let schemaElement: ERC725JSONSchema;
    if (isDynamic) {
      schemaElement = getSchemaElement(schema, keyName, dynamicKeyParts);

      // NOTE: it might be confusing to use as the output will contain other keys as the ones used
      // for the input
      return {
        key: schemaElement.key,
        name: schemaElement.name,
        value: decodeKey(schemaElement, value),
      };
    }

    schemaElement = getSchemaElement(schema, keyName);

    return {
      key: schemaElement.key,
      name: schemaElement.name,
      value: decodeKey(schemaElement, value),
    };
  };
```

```
  if (Array.isArray(data)) {
    return data.map((dataInput) => processDataInput(dataInput));
  }

  return processDataInput(data);
}
```

</file>

<file>

path: /src/lib/decodeMappingKey.test.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/lib/decodeMappingKey.test.ts

```
/*
    This file is part of @erc725/erc725.js.
    @erc725/erc725.js is free software: you can redistribute it and/or modify
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    @erc725/erc725.js is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.
    You should have received a copy of the GNU Lesser General Public License
    along with web3.js.  If not, see <http://www.gnu.org/licenses/>.
*/

/* eslint-disable no-unused-expressions */

import { expect } from 'chai';
import { decodeMappingKey } from './decodeMappingKey';

describe('decodeDynamicKeyParts', () => {
  const records = [
    {
      key: {
        name: 'MyKeyName:<address>',
        encoded:
          '0x35e6950bc8d21a1699e500000cafecafecafecafecafecafecafecafecafecafe',
      },
      dynamicKeyParts: [
        {
          type: 'address',
          value: '0xCAfEcAfeCAfECaFeCaFecaFecaFECafECafeCaFe',
        },
      ],
    },
    {
      key: {
```

```
    name: 'MyKeyName:<uint32>',
    encoded:
      '0x35e6950bc8d21a1699e50000000000000000000000000000000000000f342d33d',
  },
  dynamicKeyParts: [{ type: 'uint32', value: 4081242941 }],
},
{
  key: {
    name: 'MyKeyName:<bytes4>',
    encoded:
      '0x35e6950bc8d21a1699e50000000000000000000000000000000000000abcd1234',
  },
  dynamicKeyParts: [{ type: 'bytes4', value: 'abcd1234' }],
},
{
  key: {
    name: 'MyKeyName:<bytes32>',
    encoded:
      '0x35e6950bc8d21a1699e50000aaaabbbbccccddddeeeeffff1111222233334444',
  },
  dynamicKeyParts: [
    {
      type: 'bytes32',
      value: 'aaaabbbbccccddddeeeeffff1111222233334444',
    },
  ],
},
{
  key: {
    name: 'MyKeyName:<bool>',
    encoded:
      '0x35e6950bc8d21a1699e50000000000000000000000000000000000000000001',
  },
  dynamicKeyParts: [{ type: 'bool', value: true }],
},
{
  key: {
    name: 'MyKeyName:<bool>',
    encoded:
      '0x35e6950bc8d21a1699e50000000000000000000000000000000000000000000',
  },
  dynamicKeyParts: [{ type: 'bool', value: false }],
},
{
  key: {
    name: 'MyKeyName:MyMapName:<address>',
    encoded:
      '0x35e6950bc8d275060e3c0000cafecafecafecafecafecafecafecafecafecafe',
  },
  dynamicKeyParts: [
    {
      type: 'address',
```

```
      value: '0xCAfEcAfeCAfECaFeCaFecaFecaFECafECafeCaFe',
    },
  ],
},
{
  key: {
    name: 'MyKeyName:<bytes2>:<uint32>',
    encoded:
      '0x35e6950bc8d200000ffff0000000000000000000000000000000000000f342d33d',
  },
  dynamicKeyParts: [
    { type: 'bytes2', value: 'ffff' },
    { type: 'uint32', value: 4081242941 },
  ],
},
{
  key: {
    name: 'MyKeyName:<address>:<address>',
    encoded:
      '0x35e6950bc8d2abcdef110000cafecafecafecafecafecafecafecafecafecafe',
  },
  dynamicKeyParts: [
    {
      type: 'address',
      value: '0x0000000000000000000000000000000AbCDeF11',
    },
    {
      type: 'address',
      value: '0xCAfEcAfeCAfECaFeCaFecaFecaFECafECafeCaFe',
    },
  ],
},
{
  key: {
    name: 'MyKeyName:MyMapName:<bytes32>',
    encoded:
      '0x35e6950bc8d275060e3c0000aaaabbbbccccddddeeeeffff1111222233334444',
  },
  dynamicKeyParts: [
    {
      type: 'bytes32',
      value: 'aaaabbbbccccddddeeeeffff1111222233334444',
    },
  ],
},
{
  key: {
    name: 'MyKeyName:<bytes32>:<bool>',
    encoded:
      '0x35e6950bc8d2aaaabbbb0000000000000000000000000000000000000000000001',
  },
  dynamicKeyParts: [
```

```
      { type: 'bytes32', value: 'aaaabbbb' },
      { type: 'bool', value: true },
    ],
  },
];

it('decodes each dynamic key part', () => {
  records.forEach((record) => {
    const decodedDynamicKeyParts = decodeMappingKey(
      record.key.encoded,
      record.key.name,
    );
    expect(record.dynamicKeyParts.length).to.equal(
      decodedDynamicKeyParts.length,
    );
    record.dynamicKeyParts.forEach((keyPart, index) => {
      expect(keyPart.type).to.equal(decodedDynamicKeyParts[index].type);
      expect(keyPart.value).to.equal(decodedDynamicKeyParts[index].value);
    });
  });
});

it('decodes each dynamic key part when schema as a param', () => {
  const schema = {
    name: 'MyKeyName:<address>',
    key: '0x',
    keyType: 'Singleton',
    valueType: 'bytes',
    valueContent: 'JSONURL',
  };
  const decodedDynamicKeyParts = decodeMappingKey(
    '0x35e6950bc8d21a1699e500000cafecafecafecafecafecafecafecafecafecafe',
    schema.name,
  );
  expect(decodedDynamicKeyParts.length).to.equal(1);
  expect(decodedDynamicKeyParts[0].type).to.equal('address');
  expect(decodedDynamicKeyParts[0].value).to.equal(
    '0xCAfEcAfeCAfECaFeCaFecaFecaFECafECafeCaFe',
  );
});

it("decodes properly if only hex key without '0x'", () => {
  const decodedDynamicKeyParts = decodeMappingKey(
    '35e6950bc8d21a1699e50000000000000000000000000000000000000000001',
    'MyKeyName:<bool>',
  );
  expect(decodedDynamicKeyParts[0].value).to.equal(true);
});

it('throws if not hex encoded key', () => {
  expect(() =>
    decodeMappingKey(
```

```
      '0x3234535343fXXWGWXWDSWDAEDFAEDr5434534grdgrdggrdgdrgdgrd098594334',
      'MyKeyName:<bool>',
    ),
  ).to.throw(&amp;nbsp;Invalid encodedKey, must be a hexadecimal value&amp;nbsp;);
 });

 it('throws if incorrect length key', () => {
  expect(() =>
   decodeMappingKey(
    '0x35e6950bc8d21a1699e5000000000000000000000000000000000000000000135e6',
    'MyKeyName:<bool>',
   ),
  ).to.throw(
   &amp;nbsp;Invalid encodedKey length, key must be 32 bytes long hexadecimal value&amp;nbsp;,
  );
 });
});
```

</file>

<file>

path: /src/lib/decodeMappingKey.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/lib/decodeMappingKey.ts

```
/*
  This file is part of @erc725/erc725.js.
  @erc725/erc725.js is free software: you can redistribute it and/or modify
  it under the terms of the GNU Lesser General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.
  @erc725/erc725.js is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU Lesser General Public License for more details.
  You should have received a copy of the GNU Lesser General Public License
  along with @erc725/erc725.js.  If not, see <http://www.gnu.org/licenses/>.
*/

/**
 * @file lib/decodeMappingKey.ts
 * @author Samuel Videau <@samuel-videau>
 * @date 2022
 */

import { isHex, padLeft } from 'web3-utils';
import { decodeValueType } from './encoder';
import { ERC725JSONSchema } from '../types/ERC725JSONSchema';
import { DynamicKeyPart } from '../types/dynamicKeys';
```

```
function isDynamicKeyPart(keyPartName: string): boolean {
  return (
    keyPartName.slice(0, 1) === '<' &&
    keyPartName.slice(keyPartName.length - 1) === '>'
  );
}

/**
 * @param encodedKeyPart hashed dynamic key part
 * @param keyPartName part of a key name
 *
 * @return: the decoded value of the dynamic key part and its type (ie. 'address'; 'uint256', 'bytes32', etc)
 */
function decodeKeyPart(
  encodedKeyPart: string,
  keyPartName: string,
): DynamicKeyPart | false {
  if (!isDynamicKeyPart(keyPartName)) return false;

  let decodedKey;
  const type = keyPartName.slice(1, keyPartName.length - 1);

  if (type === 'bool')
    decodedKey = encodedKeyPart.slice(encodedKeyPart.length - 1) === '1';
  else if (type.includes('uint')) decodedKey = parseInt(encodedKeyPart, 16);
  else if (type.includes('bytes')) {
    const bytesLength = parseInt(type.replace('bytes', ''), 10) * 2;
    const sliceFrom =
      encodedKeyPart.length - bytesLength < 0
        ? 0
        : encodedKeyPart.length - bytesLength;
    decodedKey = encodedKeyPart.slice(sliceFrom);
  } else if (type === 'address') {
    // this is required if the 2nd word is an address in a MappingWithGrouping
    const leftPaddedAddress = padLeft('0x' + encodedKeyPart, 40);

    decodedKey = decodeValueType(type, leftPaddedAddress);
  } else {
    decodedKey = decodeValueType(type, encodedKeyPart);
  }

  return { type, value: decodedKey };
}

/**
 * @param keyHash hashed key with the dynamic parts
 * @param keyNameOrSchema key name of schema definitions or schema
 *
 * @return: all decoded dynamic key parts, with their type and value
 */
export function decodeMappingKey(
  keyHash: string,
```

```
  keyNameOrSchema: string | ERC725JSONSchema,
): DynamicKeyPart[] {
  let hashedKey = keyHash;
  if (hashedKey.length === 64 && hashedKey.slice(0, 2) !== '0x')
    hashedKey = '0x' + hashedKey;

  if (hashedKey.length !== 66)
    throw new Error(
       Invalid encodedKey length, key must be 32 bytes long hexadecimal value ,
    );
  if (!isHex(hashedKey.slice(2)))
    throw new Error( Invalid encodedKey, must be a hexadecimal value );

  let keyParts: string[];

  if (typeof keyNameOrSchema === 'string')
    keyParts = keyNameOrSchema.split(':');
  else keyParts = keyNameOrSchema.name.split(':');

  const dynamicParts: (DynamicKeyPart | false)[] = [];
  switch (keyParts.length) {
    case 2: // Mapping
      dynamicParts.push(decodeKeyPart(hashedKey.slice(26), keyParts[1]));
      break;

    case 3: // MappingWithGrouping
      dynamicParts.push(decodeKeyPart(hashedKey.slice(14, 22), keyParts[1]));
      dynamicParts.push(decodeKeyPart(hashedKey.slice(26), keyParts[2]));
      break;
    default:
      break;
  }

  return dynamicParts.filter((p) => p !== false) as DynamicKeyPart[];
}
```

</file>

<file>

path: /src/lib/detector.test.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/lib/detector.test.ts

```
/*
  This file is part of @erc725/erc725.js.
  @erc725/erc725.js is free software: you can redistribute it and/or modify
  it under the terms of the GNU Lesser General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.
  @erc725/erc725.js is distributed in the hope that it will be useful,
```

```
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU Lesser General Public License for more details.
   You should have received a copy of the GNU Lesser General Public License
   along with web3.js.  If not, see <http://www.gnu.org/licenses/>.
*/
/**
 * @file lib/detector.test.ts
 * @author Hugo Masclet <@Hugoo>
 * @author Felix Hildebrandt <@fhildeb>
 * @date 2022
 */

/* eslint-disable no-unused-expressions */

import { expect } from 'chai';
import * as sinon from 'sinon';
import { INTERFACE_IDS_0_12_0 } from '../constants/interfaces';

import { supportsInterface, checkPermissions } from './detector';

describe('supportsInterface', () => {
  it('it should return true if the contract supports the interface with name', async () => {
    const contractAddress = '0xcafecafecafecafecafecafecafecafecafecafe';
    const interfaceName = 'LSP0ERC725Account';

    const providerStub = { supportsInterface: sinon.stub() };

    providerStub.supportsInterface
      .withArgs(contractAddress, INTERFACE_IDS_0_12_0[interfaceName])
      .returns(Promise.resolve(true));

    const doesSupportInterface = await supportsInterface(interfaceName, {
      address: contractAddress,
      provider: providerStub,
    });

    expect(doesSupportInterface).to.be.true;
  });

  it('it should return true if the contract supports the interface with interfaceId', async () => {
    const contractAddress = '0xcafecafecafecafecafecafecafecafecafecafe';
    const interfaceId = INTERFACE_IDS_0_12_0.LSP1UniversalReceiver;

    const providerStub = { supportsInterface: sinon.stub() };

    providerStub.supportsInterface
      .withArgs(contractAddress, interfaceId)
      .returns(Promise.resolve(true));

    const doesSupportInterface = await supportsInterface(interfaceId, {
      address: contractAddress,
```

```
    provider: providerStub,
  });

  expect(doesSupportInterface).to.be.true;
 });
});

describe('checkPermissions', () => {
 describe('test with single permission', () => {
  it('should throw an error when given an invalid permission string', async () => {
   const requiredPermissions = 'INVALIDPERMISSION';
   const grantedPermissions =
    '0x000000000000000000000000000000000000000000000000000000000000ff51';
   expect(() =>
    checkPermissions(requiredPermissions, grantedPermissions),
   ).to.throw(
    'Invalid permission string. It must be a valid 32-byte hex string or a known permission name.',
   );
  });

  it('should throw an error when given an invalid 32-byte hex string', async () => {
   const requiredPermissions = '0xinvalidhexstring';
   const grantedPermissions =
    '0x000000000000000000000000000000000000000000000000000000000000ff51';
   expect(() =>
    checkPermissions(requiredPermissions, grantedPermissions),
   ).to.throw(
    'Invalid permission string. It must be a valid 32-byte hex string or a known permission name.',
   );
  });

  it('should throw an error when given an invalid grantedPermission 32-byte hex string', async () => {
   const requiredPermissions = 'CHANGEOWNER';
   const grantedPermissions = '0xinvalidgrantedpermissionhexstring';
   expect(() =>
    checkPermissions(requiredPermissions, grantedPermissions),
   ).to.throw(
    'Invalid grantedPermissions string. It must be a valid 32-byte hex string.',
   );
  });

  it('should return true when single literal permission matches granted permissions', async () => {
   const requiredPermissions = 'CHANGEOWNER';
   const grantedPermissions =
    '0x000000000000000000000000000000000000000000000000000000000000ff51';
   const result = checkPermissions(requiredPermissions, grantedPermissions);
   expect(result).to.be.true;
  });

  it('should return true when single bytes32 permission matches granted permissions', async () => {
   const requiredPermissions =
    '0x0000000000000000000000000000000000000000000000000000000000000001';
```

```
    const grantedPermissions =
      '0x0000000000000000000000000000000000000000000000000000000000000ff51';
    const result = checkPermissions(requiredPermissions, grantedPermissions);
    expect(result).to.be.true;
  });

  it('should return false when single bytes32 permission does not match granted permissions', async () => {
    const requiredPermissions =
      '0x0000000000000000000000000000000000000000000000000000000000000001';
    const grantedPermissions =
      '0x0000000000000000000000000000000000000000000000000000000000000fff2';
    const result = checkPermissions(requiredPermissions, grantedPermissions);
    expect(result).to.be.false;
  });

  it('should return false when single literal permission does not match granted permissions', async () => {
    const requiredPermissions = 'CHANGEOWNER';
    const grantedPermissions =
      '0x0000000000000000000000000000000000000000000000000000000000000fff2';
    const result = checkPermissions(requiredPermissions, grantedPermissions);
    expect(result).to.be.false;
  });
});

describe('test with multiple permissions', () => {
  it('should throw an error when given an array containing an invalid permission string', async () => {
    const requiredPermissions = ['CHANGEOWNER', 'INVALIDPERMISSION'];
    const grantedPermissions =
      '0x0000000000000000000000000000000000000000000000000000000000000ff51';
    expect(() =>
      checkPermissions(requiredPermissions, grantedPermissions),
    ).to.throw(
      'Invalid permission string. It must be a valid 32-byte hex string or a known permission name.',
    );
  });

  it('should throw an error when given an array containing an invalid 32-byte hex string', async () => {
    const requiredPermissions = ['CHANGEOWNER', '0xinvalidhexstring'];
    const grantedPermissions =
      '0x0000000000000000000000000000000000000000000000000000000000000ff51';
    expect(() =>
      checkPermissions(requiredPermissions, grantedPermissions),
    ).to.throw(
      'Invalid permission string. It must be a valid 32-byte hex string or a known permission name.',
    );
  });

  it('should return false when one of the literal permissions does not match granted permissions', async ()
=> {
    const requiredPermissions = ['EDITPERMISSIONS', 'CALL'];
    const grantedPermissions =
      '0x0000000000000000000000000000000000000000000000000000000000000ff51';
```

```
    const result = checkPermissions(requiredPermissions, grantedPermissions);
    expect(result).to.be.false;
  });

  it('should return false when one of the bytes32 permissions does not match granted permissions', async ()
=> {
    const requiredPermissions = [
      '0x0000000000000000000000000000000000000000000000000000000000000004',
      '0x0000000000000000000000000000000000000000000000000000000000000800',
    ];
    const grantedPermissions =
      '0x000000000000000000000000000000000000000000000000000000000000ff51';
    const result = checkPermissions(requiredPermissions, grantedPermissions);
    expect(result).to.be.false;
  });

  it('should return true when all the mixed literal and bytes32 permissions match granted permissions',
async () => {
    const requiredPermissions = [
      'EDITPERMISSIONS',
      '0x0000000000000000000000000000000000000000000000000000000000000800',
    ];
    const grantedPermissions =
      '0x000000000000000000000000000000000000000000000000000000000000ff54';
    const result = checkPermissions(requiredPermissions, grantedPermissions);
    expect(result).to.be.true;
  });

  it('should return false when not all multiple literal permissions match granted permissions', async () => {
    const requiredPermissions = ['CHANGEOWNER', 'CALL'];
    const grantedPermissions =
      '0x0000000000000000000000000000000000000000000000000000000000000051';
    const result = checkPermissions(requiredPermissions, grantedPermissions);
    expect(result).to.be.false;
  });

  it('should return true when all multiple literal permissions match granted permissions', async () => {
    const requiredPermissions = ['CHANGEOWNER', 'CALL'];
    const grantedPermissions =
      '0x0000000000000000000000000000000000000000000000000000000000000801';
    const result = checkPermissions(requiredPermissions, grantedPermissions);
    expect(result).to.be.true;
  });

  it('should return false when not all multiple bytes32 permissions match granted permissions', async () => {
    const requiredPermissions = [
      '0x0000000000000000000000000000000000000000000000000000000000000001',
      '0x0000000000000000000000000000000000000000000000000000000000000800',
    ];
    const grantedPermissions =
      '0x0000000000000000000000000000000000000000000000000000000000000051';
    const result = checkPermissions(requiredPermissions, grantedPermissions);
```

```
    expect(result).to.be.false;
  });

  it('should return false when not all mixed literal and bytes32 permissions match granted permissions',
async () => {
    const requiredPermissions = [
      'CHANGEOWNER',
      '0x0000000000000000000000000000000000000000000000000000000000000800',
    ];
    const grantedPermissions =
      '0x0000000000000000000000000000000000000000000000000000000000000051';
    const result = checkPermissions(requiredPermissions, grantedPermissions);
    expect(result).to.be.false;
  });
 });
});
```

</file>

<file>

path: /src/lib/detector.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/lib/detector.ts

```
/*
  This file is part of @erc725/erc725.js.
  @erc725/erc725.js is free software: you can redistribute it and/or modify
  it under the terms of the GNU Lesser General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.
  @erc725/erc725.js is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU Lesser General Public License for more details.
  You should have received a copy of the GNU Lesser General Public License
  along with @erc725/erc725.js.  If not, see <http://www.gnu.org/licenses/>.
*/

// https://docs.lukso.tech/standards/standard-detection

/**
 * @file detector.ts
 * @author Hugo Masclet <@Hugoo>
 * @author Felix Hildebrandt <@fhildeb>
 * @date 2022
 */

import { LSP6_DEFAULT_PERMISSIONS } from '../constants/constants';

import {
```

```
  AddressProviderOptions,
  INTERFACE_IDS_0_12_0,
} from '../constants/interfaces';

/**
 * Check if a smart contract address
 * supports a certain interface.
 *
 * @param {string} interfaceId  Interface ID or supported interface name.
 * @param options Object with address and RPC URL.
 * @returns {Promise<boolean>} if interface is supported.
 */
export const supportsInterface = async (
  interfaceIdOrName: string,
  options: AddressProviderOptions,
): Promise<boolean> => {
  let plainInterfaceId: string;
  if (INTERFACE_IDS_0_12_0[interfaceIdOrName]) {
    plainInterfaceId = INTERFACE_IDS_0_12_0[interfaceIdOrName];
  } else {
    plainInterfaceId = interfaceIdOrName;
  }

  try {
    return await options.provider.supportsInterface(
      options.address,
      plainInterfaceId,
    );
  } catch (error) {
    throw new Error( Error checking the interface: ${error} );
  }
};

/**
 * @notice Check if the given string is a valid 32-byte hex string.
 * @param str The string to be checked.
 * @return A boolean value indicating whether the string is a valid 32-byte hex string.
 */
function isValid32ByteHexString(str: string): boolean {
  return (
    str.startsWith('0x') &&
    str.length === 66 &&
    str
      .slice(2)
      .split('')
      .every((char) => '0123456789abcdefABCDEF'.includes(char))
  );
}

/**
 * @notice Map a permission to its corresponding bytes32 representation.
 * @param permission The permission string to be mapped.
```

```
 * @return The bytes32 representation of the permission.
 * @dev Throws an error if the input is not a known permission name or a valid 32-byte hex string.
 */
function mapPermission(permission: string): string {
  if (
    !LSP6_DEFAULT_PERMISSIONS[permission] &&
    !isValid32ByteHexString(permission)
  ) {
    throw new Error(
      'Invalid permission string. It must be a valid 32-byte hex string or a known permission name.',
    );
  }
  return LSP6_DEFAULT_PERMISSIONS[permission] || permission;
}

/**
 * @notice Check if the required permissions are included in the granted permissions.
 * @param requiredPermissions An array of required permissions or a single required permission.
 * @param grantedPermissions The granted permissions as a 32-byte hex string.
 * @return A boolean value indicating whether the required permissions are included in the granted
permissions.
 * @dev Throws an error if the grantedPermissions input is not a valid 32-byte hex string.
 */
export const checkPermissions = (
  requiredPermissions: string[] | string,
  grantedPermissions: string,
): boolean => {
  // Validate the grantedPermissions string
  if (!isValid32ByteHexString(grantedPermissions)) {
    throw new Error(
      'Invalid grantedPermissions string. It must be a valid 32-byte hex string.',
    );
  }

  // Convert requiredPermissions to an array if it's a single string
  const requiredPermissionArray: string[] = Array.isArray(requiredPermissions)
    ? requiredPermissions
    : [requiredPermissions];

  // Map the literal permissions to their bytes32 representation
  const mappedPermissionArray: string[] =
    requiredPermissionArray.map(mapPermission);

  // Perform the AND operation check for each required permission
  return mappedPermissionArray.every((requiredPermission: string) => {
    const requiredPermissionBigInt = BigInt(requiredPermission);
    const grantedPermissionsBigInt = BigInt(grantedPermissions);

    return (
      (requiredPermissionBigInt & grantedPermissionsBigInt) ===
      requiredPermissionBigInt
    );
```

```
  });
};
```

</file>

<file>

path: /src/lib/encodeKeyName.test.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/lib/encodeKeyName.test.ts

```
/*
    This file is part of @erc725/erc725.js.
    @erc725/erc725.js is free software: you can redistribute it and/or modify
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    @erc725/erc725.js is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.
    You should have received a copy of the GNU Lesser General Public License
    along with web3.js.  If not, see <http://www.gnu.org/licenses/>.
*/
/**
 * @file lib/encodeKeyName.ts
 * @author Hugo Masclet <@Hugoo>
 * @date 2021
 */

import assert from 'assert';
import { keccak256 } from 'web3-utils';

import {
  encodeDynamicKeyPart,
  encodeKeyName,
  generateDynamicKeyName,
  isDynamicKeyName,
} from './encodeKeyName';

describe('encodeKeyName', () => {
  const testCases: {
    keyName: string;
    expectedKey: string;
    dynamicKeyParts?: string | string[];
  }[] = [
    {
      keyName: 'MyKeyName',
      expectedKey: keccak256('MyKeyName'),
    },
    {
```

```
  keyName: 'LSP3IssuedAssets[]',
  expectedKey:
    '0x3a47ab5bd3a594c3a8995f8fa58d0876c96819ca4516bd76100c92462f2f9dc0',
},
{
  keyName: 'SupportedStandards:LSP3Profile',
  expectedKey:
    '0xeafec4d89fa9619884b600005ef83ad9559033e6e941db7d7c495acdce616347',
},
{
  keyName: 'MyCoolAddress:0xcafecafecafecafecafecafecafecafecafecafe',
  expectedKey:
    '0x22496f48a493035f0ab40000cafecafecafecafecafecafecafecafecafe',
},
{
  keyName: 'MyCoolAddress:cafecafecafecafecafecafecafecafecafecafe',
  expectedKey:
    '0x22496f48a493035f0ab40000cafecafecafecafecafecafecafecafecafe',
},
{
  keyName: 'LSP12IssuedAssetsMap:b74a88C43BCf691bd7A851f6603cb1868f6fc147',
  expectedKey:
    '0x74ac2555c10b9349e78f0000b74a88c43bcf691bd7a851f6603cb1868f6fc147',
},
{
  keyName:
    'AddressPermissions:Permissions:cafecafecafecafecafecafecafecafecafecafe',
  expectedKey:
    '0x4b80742de2bf82acb3630000cafecafecafecafecafecafecafecafecafe',
},
{
  keyName:
    'AddressPermissions:Permissions:0xcafecafecafecafecafecafecafecafecafecafe',
  expectedKey:
    '0x4b80742de2bf82acb3630000cafecafecafecafecafecafecafecafecafe',
},
// DYNAMIC KEYS
// Mapping
{
  keyName: 'MyDynamicKey:<address>',
  expectedKey:
    '0xd1b2917d26eeeaad5b980000cafecafecafecafecafecafecafecafecafe',
  // |-keccak256 bytes10--||--||---bytes20: the address----------|
  dynamicKeyParts: ['0xcafecafecafecafecafecafecafecafecafecafe'], // as array
},
{
  keyName: 'MyDynamicKey:<address>',
  expectedKey:
    '0xd1b2917d26eeeaad5b980000cafecafecafecafecafecafecafecafecafe',
  dynamicKeyParts: '0xcafecafecafecafecafecafecafecafecafecafe', // as string
},
{
```

```
  keyName: 'MyDynamicKey:<address>',
  expectedKey:
    '0xd1b2917d26eeeaad5b980000cafecafecafecafecafecafecafecafecafecafe',
  dynamicKeyParts: 'cafecafecafecafecafecafecafecafecafecafe', // without 0x prefix
},
{
  keyName: 'MyKeyName:<string>',
  expectedKey:
    '0x35e6950bc8d21a1699e5000075060e3cd7d404500e94d415fb5992ced9ad8f058',
  // |--keccak256 bytes12-ll--ll---bytes20: keccak256()---------------|
  dynamicKeyParts: 'MyMapName',
},
{
  keyName: 'MyKeyName:<uint32>',
  expectedKey:
    '0x35e6950bc8d21a1699e50000000000000000000000000000000000f342d33d',
  // |--keccak256 bytes12-ll--ll---bytes20: keccak256()---------------|
  dynamicKeyParts: '4081242941',
},
{
  keyName: 'MyKeyName:<uint32>',
  expectedKey:
    '0x35e6950bc8d21a1699e50000000000000000000000000000000000f342d33d',
  // |--keccak256 bytes12-ll--ll---bytes20: keccak256()---------------|
  dynamicKeyParts: '0xf342d33d',
},
{
  keyName: 'MyKeyName:<bytes4>',
  expectedKey:
    '0x35e6950bc8d21a1699e50000000000000000000000000000000000abcd1234',
  // |--keccak256 bytes12-ll--ll---bytes20: keccak256()---------------|
  dynamicKeyParts: '0xabcd1234',
},
{
  keyName: 'MyKeyName:<bytes32>',
  expectedKey:
    '0x35e6950bc8d21a1699e50000aaaabbbbccccddddeeeeffff1111222233334444',
  // |--keccak256 bytes12-ll--ll---bytes20: keccak256()---------------|
  dynamicKeyParts:
    '0xaaaabbbbccccddddeeeeffff11112222333344445555666677778888999aaaa',
},
{
  keyName: 'MyKeyName:<bool>',
  expectedKey:
    '0x35e6950bc8d21a1699e50000000000000000000000000000000000000001',
  // |--keccak256 bytes12-ll--ll-----------------------------------|
  dynamicKeyParts: 'true',
},
{
  keyName: 'MyKeyName:<bool>',
  expectedKey:
    '0x35e6950bc8d21a1699e50000000000000000000000000000000000000000',
```

```
    // |--keccak256 bytes12-||--||-----------------------------------|
    dynamicKeyParts: 'false',
  },
  // MappingWithGrouping
  {
    keyName: 'MyKeyName:MyMapName:<address>',
    expectedKey:
      '0x35e6950bc8d275060e3c0000cafecafecafecafecafecafecafecafecafecafe',
    // |-- bytes6 --||------||--||---------- bytes20 ----------------|
    dynamicKeyParts: '0xcafecafecafecafecafecafecafecafecafecafe',
  },
  {
    keyName: 'MyKeyName:MyMapName:<address>',
    expectedKey:
      '0x35e6950bc8d275060e3c00004c6f947ae67f572afa4ae07309477de7c874f95ef',
    // |-- bytes6 --||------||--||---------- bytes20 ----------------|
    dynamicKeyParts: '0x4c6f947Ae67F572afa4ae07309477DE7C874F95Ef', // address is checksummed ->
expected key should be lowercase
  },
  {
    keyName: 'MyKeyName:<bytes2>:<uint32>',
    expectedKey:
      '0x35e6950bc8d200000ffff000000000000000000000000000000000000f342d33d',
    // |-- bytes6 --||------||--||---------- bytes20 ----------------|
    dynamicKeyParts: ['ffff', '4081242941'],
  },
  {
    keyName: 'MyKeyName:<bytes2>:<uint32>',
    expectedKey:
      '0x35e6950bc8d200000ffff000000000000000000000000000000000000f342d33d',
    // |-- bytes6 --||------||--||---------- bytes20 ----------------|
    dynamicKeyParts: ['ffff', '0xf342d33d'],
  },
  {
    keyName: 'MyKeyName:<address>:<address>',
    expectedKey:
      '0x35e6950bc8d2abcdef110000cafecafecafecafecafecafecafecafecafecafe',
    // |-- bytes6 --||------||--||---------- bytes20 ----------------|
    dynamicKeyParts: [
      '0xabcdef11abcdef11abcdef11abcdef11ffffffff',
      '0xcafecafecafecafecafecafecafecafecafecafe',
    ],
  },
  {
    keyName: 'MyKeyName:MyMapName:<bytes32>',
    expectedKey:
      '0x35e6950bc8d275060e3c0000aaaabbbbccccddddeeeeffff1111222233334444',
    // |-- bytes6 --||------||--||---------- bytes20 ----------------|
    dynamicKeyParts: [
      '0xaaaabbbbccccddddeeeeffff11112222333344445555666677778888999aaaa',
    ],
  },
```

```
  {
    keyName: 'MyKeyName:<bytes32>:<bool>',
    expectedKey:
      '0x35e6950bc8d2aaaabbbb0000000000000000000000000000000000000000000001',
    // |-- bytes6 --||------||--||---------- bytes20 ----------------|
    dynamicKeyParts: [
      '0xaaaabbbbccccddddeeeeffff11112222333344445555666677778888999aaaa',
      'true',
    ],
  },
  {
    keyName: 'MyKeyName:<bytes32>:MyMapName',
    expectedKey:
      '0x35e6950bc8d2aaaabbbb0000750600e3cd7d40450e94d415fb5992ced9ad8f058',
    // |-- bytes6 --||------||--||---------- bytes20 ----------------|
    dynamicKeyParts: [
      '0xaaaabbbbccccddddeeeeffff11112222333344445555666677778888999aaaa',
    ],
  },
];

testCases.forEach((testCase) => {
  it(&amp;nbsp;encodes ${testCase.keyName} key name correctly&amp;nbsp;;, () => {
    assert.deepStrictEqual(
      encodeKeyName(testCase.keyName, testCase.dynamicKeyParts),
      testCase.expectedKey,
    );
  });
});

it('throws if trying to encode a dynamic key without any variable', () => {
  assert.throws(() => encodeKeyName('MyDynamicKey:<address>'));
});
it('throws if trying to encode a dynamic key of type: Mapping with more or less than 1 variable', () => {
  assert.throws(() => encodeKeyName('MyDynamicKey:<address>', []));
  assert.throws(() =>
    encodeKeyName('MyDynamicKey:<address>', [
      '0xcafecafecafecafecafecafecafecafecafecafe',
      'extraVariable',
    ]),
  );
});

it('throws if trying to encode a dynamic key of type: MappingWithGrouping with more wrong number of
variables', () => {
  assert.throws(() => encodeKeyName('MyDynamicKey:Grouping:<address>', []));
  assert.throws(() =>
    encodeKeyName('MyDynamicKey:Group:<address>', [
      '0xcafecafecafecafecafecafecafecafecafecafe',
      'extraVariable',
    ]),
  );
```

```
    assert.throws(() =>
      encodeKeyName('MyDynamicKey:<string>:<address>', ['variable1']),
    );
    assert.throws(() =>
      encodeKeyName(
        'MyDynamicKey:<string>:cafecafecafecafecafecafecafecafecafecafe',
        ['variable1', 'variable2'],
      ),
    );
  });

  it('throws if a dynamic key is given without dynamicKeyParts', () => {
    assert.throws(() => encodeKeyName('MyDynamicKey:<address>'));
  });
});

describe('isDynamicKeyName', () => {
  const testCases = [
    {
      keyName: 'AddressPermissions[]',
      isDynamicKeyName: false,
    },
    {
      keyName: 'LSP4TokenSymbol',
      isDynamicKeyName: false,
    },
    {
      keyName: 'LSP5ReceivedAssetsMap:<address>',
      isDynamicKeyName: true,
    },
    {
      keyName: 'AddressPermissions:AllowedStandards:<address>',
      isDynamicKeyName: true,
    },
    {
      keyName: 'AddressPermissions:<string>:<address>',
      isDynamicKeyName: true,
    },
    {
      keyName: '<string>',
      isDynamicKeyName: true,
    },
  ];

  testCases.forEach((testCase) => {
    it( detects ${
      testCase.isDynamicKeyName ? 'dynamic' : 'non-dynamic'
    } key name: ${testCase.keyName} correctly , () => {
      assert.deepStrictEqual(
        isDynamicKeyName(testCase.keyName),
        testCase.isDynamicKeyName,
      );
```

```
    });
  });
});

describe('encodeDynamicKeyPart', () => {
  const testCases = [
    {
      type: '<string>',
      value: 'HelloHowAreYou',
      bytes: 20,
      expectedEncoding: '81ca1f336033f64c1b23b11b227c3ba7e87f3f73',
    },
    {
      type: '<bool>',
      value: 'true',
      bytes: 4,
      expectedEncoding: '00000001',
    },
    {
      type: '<bool>',
      value: 'false',
      bytes: 4,
      expectedEncoding: '00000000',
    },
    {
      type: '<address>',
      value: '0x7f268357a8c2552623316e2562d90e642bb538e5',
      bytes: 5,
      expectedEncoding: '7f268357a8', // right-cut if larger than bytes
    },
    {
      type: '<address>',
      value: '0x1FdCD4dD0E164bCbd32DbF983c9903F4eD10356d',
      bytes: 5,
      expectedEncoding: '1FdCD4dD0E'.toLowerCase(), // should return lowercase
    },
    {
      type: '<address>',
      value: '7f268357a8c2552623316e2562d90e642bb538e5',
      bytes: 21,
      expectedEncoding: '007f268357a8c2552623316e2562d90e642bb538e5', // left padded
    },
    {
      type: '<uint32>',
      value: '4081242941',
      bytes: 20,
      expectedEncoding: '00000000000000000000000000000000f342d33d', // left padded
    },
    {
      type: '<uint32>',
      value: '0xf342d33d',
      bytes: 20,
```

```
    expectedEncoding: '00000000000000000000000000000000f342d33d', // left padded
  },
  {
    type: '<uint32>',
    value: '4081242941',
    bytes: 2,
    expectedEncoding: 'd33d', // left cut
  },
  {
    type: '<uint32>',
    value: '0xf342d33d',
    bytes: 2,
    expectedEncoding: 'd33d', // left cut
  },
  // TODO: add intM cases
  {
    type: '<bytes8>',
    value: '0xd1b2917d26eeeaad',
    bytes: 12,
    expectedEncoding: '00000000d1b2917d26eeeaad', // left padded
  },
  {
    type: '<bytes8>',
    value: 'd1b2917d26eeeaad', // test without 0x prefix
    bytes: 12,
    expectedEncoding: '00000000d1b2917d26eeeaad', // left padded
  },
  {
    type: '<bytes8>',
    value: 'd1b2917d26eeeaad',
    bytes: 4,
    expectedEncoding: 'd1b2917d', // right cut
  },
];

testCases.forEach((testCase) => {
  it( encodes: ${testCase.value} of type: ${testCase.type} correctly , () => {
    assert.deepStrictEqual(
      encodeDynamicKeyPart(testCase.type, testCase.value, testCase.bytes),
      testCase.expectedEncoding,
    );
  });
});

it('throws if <bytesM> is called with non hex values', () => {
  assert.throws(() =>
    encodeDynamicKeyPart('<bytes8>', 'thisisnotanhexstr', 20),
  );
});
it('throws if <bytesM> is called with wrong number of bytes', () => {
  assert.throws(() =>
    encodeDynamicKeyPart('<bytes8>', '0xd1b2917d26eeeaad1234', 20),
```

```
    );
  });
});

describe('generateDynamicKeyName', () => {
  const testCases = [
    {
      keyName: 'MyKey:<string>',
      dynamicKeyParts: 'HelloHowAreYou',
      expectedKeyName: 'MyKey:HelloHowAreYou',
    },
    {
      keyName: 'MyKey:Grouping:<string>',
      dynamicKeyParts: 'HelloHowAreYou',
      expectedKeyName: 'MyKey:Grouping:HelloHowAreYou',
    },

    {
      keyName: 'MyKey:<bytes4>:<address>',
      dynamicKeyParts: [
        '0x11223344',
        '0x2ab3903c6e5815f4bc2a95b7f3b22b6a289bacac',
      ],
      expectedKeyName:
        'MyKey:11223344:2ab3903c6e5815f4bc2a95b7f3b22b6a289bacac',
    },
    {
      keyName: 'Addresses:<address>',
      dynamicKeyParts: [
        '2ab3903c6e5815f4bc2a95b7f3b22b6a289bacac', // without 0x in the address
      ],
      expectedKeyName: 'Addresses:2ab3903c6e5815f4bc2a95b7f3b22b6a289bacac',
    },
  ];

  testCases.forEach((testCase) => {
    it(&amp;nbsp;generates key name: ${testCase.keyName} correctly&amp;nbsp;;, () => {
      assert.deepStrictEqual(
        generateDynamicKeyName(testCase.keyName, testCase.dynamicKeyParts),
        testCase.expectedKeyName,
      );
    });
  });

  it('throws if encoding with wrong number of dynamic values', () => {
    assert.throws(() =>
      generateDynamicKeyName('Addresses:<bytes4>:<address>', '0x11223344'),
    );
  });
});
```

</file>

<file>

path: /src/lib/encodeKeyName.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/lib/encodeKeyName.ts

```
/*
    This file is part of @erc725/erc725.js.
    @erc725/erc725.js is free software: you can redistribute it and/or modify
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    @erc725/erc725.js is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.
    You should have received a copy of the GNU Lesser General Public License
    along with @erc725/erc725.js.  If not, see <http://www.gnu.org/licenses/>.
*/
/**
 * @file lib/encodeKeyName.ts
 * @author Hugo Masclet <@Hugoo>
 * @date 2022
 */

import {
  isAddress,
  isHex,
  keccak256,
  leftPad,
  numberToHex,
  padLeft,
} from 'web3-utils';

import { guessKeyTypeFromKeyName } from './utils';
import { DynamicKeyParts } from '../types/dynamicKeys';

// https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-2-ERC725YJSONSchema.md#mapping

const dynamicTypes = ['<string>', '<address>', '<bool>'];

// https://docs.soliditylang.org/en/v0.8.14/abi-spec.html#types
const dynamicTypesRegex = /<(uintintlbytes)(\d+)>/;

/**
 *
 * @param type <string>, <uintM>, <intM>, <bool>, <bytesM>, <address>.
 * @param value
 * @param bytes the number of bytes to keep / padding
 */
```

```
export const encodeDynamicKeyPart = (
  type: string,
  value: string,
  bytes: number,
) => {
  let baseType = '';
  let size = 0;

  if (dynamicTypes.includes(type)) {
    baseType = type.slice(1, -1);
  } else {
    const regexMatch = type.match(dynamicTypesRegex);

    if (!regexMatch) {
      throw new Error( Dynamic key: ${type} is not supported );
    }

    // eslint-disable-next-line prefer-destructuring
    baseType = regexMatch[1];
    size = parseInt(regexMatch[2], 10);
  }

  switch (baseType) {
    case 'string':
      return keccak256(value).slice(2, 2 + bytes * 2);
    case 'bool': {
      if (value !== 'true' && value !== 'false') {
        throw new Error(
           Wrong value: ${value} for dynamic key with type: <bool>. Expected "true" or "false". ,
        );
      }
      return leftPad(+(value === 'true'), bytes * 2).slice(2);
    }
    case 'address': {
      if (!isAddress(value)) {
        throw new Error(
           Wrong value: ${value} for dynamic key with type: <address>. Value is not an address. ,
        );
      }

      if (bytes > 20) {
        return leftPad(value.replace('0x', ''), bytes * 2).toLowerCase();
      }

      return value
        .replace('0x', '')
        .slice(0, bytes * 2)
        .toLowerCase();
    }
    case 'uint': {
      if (size > 256 || size % 8 !== 0) {
        throw new Error(
```

```
         Wrong dynamic key type: ${type}. 0 < M <= 256, M % 8 == 0. Got: ${size}. ,
      );
    }

    // NOTE: we could verify if the number given is not too big for the given size.
    // e.g.: uint8 max value is 255, uint16 is 65535...

    const hexNumber = numberToHex(value).slice(2);
    if (hexNumber.length <= bytes * 2) {
      return padLeft(hexNumber, bytes * 2);
    }

    return hexNumber.slice(-bytes * 2);
  }
  case 'int':
    // TODO:
    throw new Error('The encoding of <intM> has not been implemented yet.');
  case 'bytes': {
    const valueWithoutPrefix = value.replace('0x', '');
    if (valueWithoutPrefix.length !== size * 2) {
      throw new Error(
         Wrong value: ${value} for dynamic key with type: ${type}. Value is not ${size} bytes
long. ,
      );
    }

    if (!isHex(valueWithoutPrefix)) {
      throw new Error(
         Wrong value: ${value} for dynamic key with type: ${type}. Value is not in hex. ,
      );
    }

    if (valueWithoutPrefix.length > bytes * 2) {
      return valueWithoutPrefix.slice(0, bytes * 2); // right cut
    }

    return leftPad(valueWithoutPrefix, bytes * 2).toLowerCase();
  }
  default:
    throw new Error( Dynamic key: ${type} is not supported );
 }
};

export function isDynamicKeyName(name: string) {
 const keyNameParts = name.split(':');

 for (let i = 0; i < keyNameParts.length; i++) {
  if (
    dynamicTypes.includes(keyNameParts[i]) ||
    keyNameParts[i].match(dynamicTypesRegex)
  ) {
    return true;
```

```
    }
  }
  return false;
}

/**
 * Encodes a MappingWithGrouping with dynamic values, according to LSP-2 ERC725YJSONSchema.
 * https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-2-ERC725YJSONSchema.md#mapping
 * bytes10:bytes2(0):bytes20
 *
 *
 * @param name Ex: MyKeyName:<address>
 * @param dynamicKeyParts ['0xcafecafecafecafecafecafecafecafecafecafe']
 * @returns the encoded key
 */
const encodeDynamicMapping = (name: string, dynamicKeyParts: string[]) => {
  if (dynamicKeyParts.length !== 1) {
    throw new Error(
       Dynamic key of type: Mapping expects exactly 1 variable. Got: ${dynamicKeyParts.length}
(${dynamicKeyParts}) ,
    );
  }

  const keyNameSplit = name.split(':'); // LSP5ReceivedAssetsMap:<address>

  const encodedKey = keccak256(keyNameSplit[0]).slice(0, 22);

  return  ${encodedKey}0000${encodeDynamicKeyPart(
    keyNameSplit[1],
    dynamicKeyParts[0],
    20,
  )} ;
};

/**
 * Encodes a MappingWithGrouping with dynamic values, according to LSP-2 ERC725YJSONSchema.
 * https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-2-
ERC725YJSONSchema.md#mappingwithgrouping
 * bytes6:bytes4:bytes2(0):bytes20
 *
 * @param name
 * @param dynamicKeyParts
 * @returns the encoded key
 */
const encodeDynamicMappingWithGrouping = (
  name: string,
  dynamicKeyParts: string[],
) => {
  const keyNameSplit = name.split(':'); // MyGroup:<string>:<address>

  let numberOfVariables = 0;
  if (isDynamicKeyName(keyNameSplit[1])) {
```

```
  numberOfVariables += 1;
 }
 if (isDynamicKeyName(keyNameSplit[2])) {
  numberOfVariables += 1;
 }

 if (numberOfVariables !== dynamicKeyParts.length) {
  throw new Error(
    Can not encode dynamic key of type: MappingWithGrouping. Wrong number of arguments.
Expects exactly ${numberOfVariables} variable(s), got: ${dynamicKeyParts.length}
(${dynamicKeyParts}) ,
  );
 }

 const firstPart = keccak256(keyNameSplit[0]).slice(0, 14);

 let secondPart = '';
 if (isDynamicKeyName(keyNameSplit[1])) {
  secondPart = encodeDynamicKeyPart(keyNameSplit[1], dynamicKeyParts[0], 4);
 } else {
  secondPart = keccak256(keyNameSplit[1]).slice(2, 2 + 4 * 2);
 }

 let lastPart = '';
 if (isDynamicKeyName(keyNameSplit[2])) {
  lastPart = encodeDynamicKeyPart(
   keyNameSplit[2],
   dynamicKeyParts[dynamicKeyParts.length - 1],
   20,
  );
 } else {
  lastPart = keccak256(keyNameSplit[2]).slice(2, 2 + 20 * 2);
 }

 return  ${firstPart}${secondPart}0000${lastPart} ;
};

function encodeDynamicKeyName(
 name: string,
 dynamicKeyParts?: DynamicKeyParts,
): string {
 if (!dynamicKeyParts) {
  throw new Error(
    Can't encode dynamic key name: ${name} without dynamicKeyParts ,
  );
 }

 const dynamicKeyPartsArray =
  typeof dynamicKeyParts === 'string' ? [dynamicKeyParts] : dynamicKeyParts;

 const keyType = guessKeyTypeFromKeyName(name);
```

```
  switch (keyType) {
    case 'Mapping':
      return encodeDynamicMapping(name, dynamicKeyPartsArray);
    case 'MappingWithGrouping':
      return encodeDynamicMappingWithGrouping(name, dynamicKeyPartsArray);
    default:
      throw new Error(
         Could not encode dynamic key: ${name} of type: ${keyType} ,
      );
  }
}

/**
 *
 * @param name the schema element name.
 * @param dynamicKeyParts
 *
 * @return the name of the key encoded as per specifications.
 */
export function encodeKeyName(name: string, dynamicKeyParts?: DynamicKeyParts) {
  if (isDynamicKeyName(name)) {
    return encodeDynamicKeyName(name, dynamicKeyParts);
  }

  const keyType = guessKeyTypeFromKeyName(name);

  switch (keyType) {
    case 'MappingWithGrouping': {
      const keyNameSplit = name.split(':');

      return encodeDynamicMappingWithGrouping(
         ${keyNameSplit[0]}:<string>:<address> ,
        [keyNameSplit[1], keyNameSplit[2]],
      );
    }

    case 'Mapping': {
      const keyNameSplit = name.split(':');
      if (isAddress(keyNameSplit[1])) {
        return encodeDynamicMapping( ${keyNameSplit[0]}:<address> , [
          keyNameSplit[1],
        ]);
      }

      return encodeDynamicMapping( ${keyNameSplit[0]}:<string> , [
        keyNameSplit[1],
      ]);
    }
    case 'Array': // Warning: this can not correctly encode subsequent keys of array, only the initial Array key
will work
    case 'Singleton':
      return keccak256(name);
```

```
    default:
      return keccak256(name);
  }
}

export const generateDynamicKeyName = (
  name: string,
  dynamicKeyParts: DynamicKeyParts,
) => {
  let dynamicKeyPartsIndex = 0;
  const dynamicKeyPartsArray =
    typeof dynamicKeyParts === 'string' ? [dynamicKeyParts] : dynamicKeyParts;

  return name
    .split(':')
    .map((keyNamePart) => {
      if (!isDynamicKeyName(keyNamePart)) {
        return keyNamePart;
      }

      // We could add more checks to make sure the variable in dynamicKeyParts[i] is matching the type in the keyName
      if (!dynamicKeyPartsArray[dynamicKeyPartsIndex]) {
        throw new Error(
           Can not generate key name: ${name}. Missing/not enough dynamicKeyParts: ${dynamicKeyPartsArray} ,
        );
      }

      const dynamicKeyPart = dynamicKeyPartsArray[dynamicKeyPartsIndex];

      if (keyNamePart === '<address>') {
        if (!isAddress(dynamicKeyPart)) {
          throw new Error(
             Dynamic key is expecting an <address> but got: ${dynamicKeyPart} ,
          );
        }
      }
      // Add more checks for bytes, etc.

      dynamicKeyPartsIndex += 1;

      return dynamicKeyPart.replace('0x', '');
    })
    .join(':');
};
```

</file>

<file>

path: /src/lib/encoder.test.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/lib/encoder.test.ts

```
/*
    This file is part of @erc725/erc725.js.
    @erc725/erc725.js is free software: you can redistribute it and/or modify
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    @erc725/erc725.js is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.
    You should have received a copy of the GNU Lesser General Public License
    along with web3.js.  If not, see <http://www.gnu.org/licenses/>.
*/

/* eslint-disable no-unused-expressions */

import { expect, assert } from 'chai';

import { keccak256, utf8ToHex, stripHexPrefix, toBN, toHex } from 'web3-utils';
import {
  valueContentEncodingMap,
  encodeValueType,
  decodeValueType,
  encodeValueContent,
  decodeValueContent,
} from './encoder';
import {
  SUPPORTED_VERIFICATION_METHOD_HASHES,
  SUPPORTED_VERIFICATION_METHOD_STRINGS,
} from '../constants/constants';
import { JSONURLDataToEncode, URLDataWithHash } from '../types';

describe('encoder', () => {
  describe('valueType', () => {
    describe(' bool / boolean  type', () => {
      const validTestCases = [
        {
          valueType: 'bool',
          decodedValue: true,
          encodedValue: '0x01',
        },
        {
          valueType: 'bool',
          decodedValue: false,
          encodedValue: '0x00',
        },
        {
```

```
      valueType: 'boolean', // allow to specify "boolean"
      decodedValue: true,
      encodedValue: '0x01',
    },
    {
      valueType: 'boolean', // allow to specify "boolean"
      decodedValue: false,
      encodedValue: '0x00',
    },
  ];

  validTestCases.forEach((testCase) => {
    it(&amp;nbsp;encodes/decodes: ${testCase.decodedValue} as ${testCase.valueType}&amp;nbsp;, () => {
      const encodedValue = encodeValueType(
        testCase.valueType,
        testCase.decodedValue,
      );

      assert.deepStrictEqual(encodedValue, testCase.encodedValue);
      assert.deepStrictEqual(
        decodeValueType(testCase.valueType, encodedValue),
        testCase.decodedValue,
      );
    });
  });
});

describe('&amp;nbsp;bytes4&amp;nbsp; type', () => {
  const validTestCases = [
    {
      valueType: 'bytes4',
      decodedValue: '0x13370000',
      encodedValue: '0x13370000',
    },
    {
      valueType: 'bytes4',
      decodedValue: '0xcafecafe',
      encodedValue: '0xcafecafe',
    },
  ];

  validTestCases.forEach((testCase) => {
    it(&amp;nbsp;encodes/decodes: ${testCase.decodedValue} as ${testCase.valueType}&amp;nbsp;, () => {
      const encodedValue = encodeValueType(
        testCase.valueType,
        testCase.decodedValue,
      );

      assert.deepStrictEqual(encodedValue, testCase.encodedValue);
      assert.deepStrictEqual(
        decodeValueType(testCase.valueType, encodedValue),
        testCase.decodedValue,
```

```
        );
      });
    });

    const errorEncodingTestCases = [
      {
        valueType: 'bytes4',
        input: '0x0000000000001337', // more than 4 bytes
      },
      {
        valueType: 'bytes4',
        input: '0xcafecafecafecafe', // more than 4 bytes
      },
      {
        valueType: 'bytes4',
        input: 'hello there', // string input that converts to more than 4 bytes in hex
      },
      {
        valueType: 'bytes4',
        input: 2 ** (8 * 4), // max number ( uint32 ), does not fit in 4 bytes (= 0x0100000000)
      },
    ];

    errorEncodingTestCases.forEach((testCase) => {
      it( should throw when trying to encode ${testCase.input} as ${testCase.valueType} , async
() => {
        assert.throws(() =>
          encodeValueType(testCase.valueType, testCase.input),
        );
      });
    });

    // these cases are not symetric. The input is converted + encoded.
    // When decoding, we do not get the same input back, but its bytes4 hex representation
    const oneWayEncodingTestCases = [
      {
        valueType: 'bytes4',
        input: 'week', // 4 letter word (= 4 bytes),
        encodedValue: '0x7765656b', // utf8-encoded characters
        decodedValue: '0x7765656b',
      },
      {
        valueType: 'bytes4',
        input: 1122334455,
        encodedValue: '0x42e576f7', // number converted to hex + right padded
        decodedValue: '0x42e576f7',
      },
    ];

    oneWayEncodingTestCases.forEach((testCase) => {
      it( encodes one way \ input\  = ${testCase.input} as ${testCase.valueType}, but does
not decode back as the same input , async () => {
```

```
        const encodedValue = encodeValueType(
          testCase.valueType,
          testCase.input,
        );

        assert.deepStrictEqual(encodedValue, testCase.encodedValue);
        assert.deepStrictEqual(
          decodeValueType(testCase.valueType, encodedValue),
          testCase.decodedValue,
        );
      });
    });

    // these cases are not symetric and right pad the value
    const rightPaddedTestCases = [
      {
        valueType: 'bytes4',
        input: '0xf00d',
        encodedValue: '0xf00d0000', // pad on the right with 2x 0x00 bytes
        decodedValue: '0xf00d0000',
      },
      {
        valueType: 'bytes4',
        input: 'yes', // convert to utf8 hex + pad on the right with 1x 0x00 byte
        encodedValue: '0x79657300',
        decodedValue: '0x79657300',
      },
    ];

    rightPaddedTestCases.forEach((testCase) => {
      it(&amp;nbsp;encodes + right pad \&amp;nbsp;input\&amp;nbsp; = ${testCase.input} as ${testCase.valueType}
padded on the right with \&amp;nbsp;00\&amp;nbsp;s&amp;nbsp;, async () => {
        const encodedValue = encodeValueType(
          testCase.valueType,
          testCase.input,
        );

        assert.deepStrictEqual(encodedValue, testCase.encodedValue);
        assert.deepStrictEqual(
          decodeValueType(testCase.valueType, encodedValue),
          testCase.decodedValue,
        );
      });
    });
  });

  describe('&amp;nbsp;bytes32&amp;nbsp; type', () => {
    const validTestCases = [
      {
        valueType: 'bytes32',
        decodedValue:
          '0x1337000000000000000000000000000000000000000000000000000000000000',
```

```
    encodedValue:
      '0x1337000000000000000000000000000000000000000000000000000000000000',
  },
  {
    valueType: 'bytes32',
    decodedValue:
      '0xcafecafecafecafecafecafecafecafecafecafecafecafecafecafecafecafe',
    encodedValue:
      '0xcafecafecafecafecafecafecafecafecafecafecafecafecafecafecafecafe',
  },
];

validTestCases.forEach((testCase) => {
  it( encodes/decodes: ${testCase.decodedValue} as ${testCase.valueType} , () => {
    const encodedValue = encodeValueType(
      testCase.valueType,
      testCase.decodedValue,
    );

    assert.deepStrictEqual(encodedValue, testCase.encodedValue);
    assert.deepStrictEqual(
      decodeValueType(testCase.valueType, encodedValue),
      testCase.decodedValue,
    );
  });
});

const errorEncodingTestCases = [
  {
    valueType: 'bytes32',
    // too many bytes (= 40 bytes)
    input:
      '0xcafecafecafecafecafecafecafecafecafecafecafecafecafecafecafecafecafecafe',
  },
  {
    valueType: 'bytes32',
    // more than 32 characters, does not fit
    input: 'This is a very long sentence that is more than 32 bytes.',
  },
  {
    valueType: 'bytes32',
    // over the max uint256 allowed, does not fit in 32 bytes
    input: toHex(
      toBN(
        '0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff' +
          1,
      ),
    ),
  },
];

errorEncodingTestCases.forEach((testCase) => {
```

```
      it(&amp;nbsp;should throw when trying to encode ${testCase.input} as ${testCase.valueType}&amp;nbsp;, async
() => {
        assert.throws(() =>
          encodeValueType(testCase.valueType, testCase.input),
        );
      });
    });

    const oneWayEncodingTestCases = [
      {
        valueType: 'bytes32',
        input: 'This sentence is 32 bytes long !',
        decodedValue:
          '0x546869732073656e74656e636520697320333220627974657320c6f6e672021',
        encodedValue:
          '0x546869732073656e74656e636520697320333220627974657320c6f6e672021',
      },
      {
        valueType: 'bytes32',
        input: 12345,
        decodedValue:
          '0x3039000000000000000000000000000000000000000000000000000000000000',
        encodedValue:
          '0x3039000000000000000000000000000000000000000000000000000000000000',
      },
    ];

    oneWayEncodingTestCases.forEach((testCase) => {
      it(&amp;nbsp;encodes one way \&amp;nbsp;input\&amp;nbsp; = ${testCase.input} as ${testCase.valueType}, but does
not decode back as the same input&amp;nbsp;, async () => {
        const encodedValue = encodeValueType(
          testCase.valueType,
          testCase.input,
        );

        assert.deepStrictEqual(encodedValue, testCase.encodedValue);
        assert.deepStrictEqual(
          decodeValueType(testCase.valueType, encodedValue),
          testCase.decodedValue,
        );
      });
    });

    // these cases are not symetric and right pad the value
    const rightPaddedTestCases = [
      {
        valueType: 'bytes32',
        input: '0xcafecafe',
        decodedValue:
          '0xcafecafe0000000000000000000000000000000000000000000000000000000',
        encodedValue:
          '0xcafecafe0000000000000000000000000000000000000000000000000000000',
```

```
      },
      {
        valueType: 'bytes32',
        input: 'hello world!',
        decodedValue:
          '0x68656c6c6f20776f726c6421000000000000000000000000000000000000000000',
        encodedValue:
          '0x68656c6c6f20776f726c6421000000000000000000000000000000000000000000',
      },
    ];

    rightPaddedTestCases.forEach((testCase) => {
      it( encodes + right pad \ input\  = ${testCase.input} as ${testCase.valueType}
padded on the right with \ 00\ s , async () => {
        const encodedValue = encodeValueType(
          testCase.valueType,
          testCase.input,
        );

        assert.deepStrictEqual(encodedValue, testCase.encodedValue);
        assert.deepStrictEqual(
          decodeValueType(testCase.valueType, encodedValue),
          testCase.decodedValue,
        );
      });
    });
  });

  describe(' uint128  type', () => {
    const validTestCases = [
      {
        valueType: 'uint128',
        decodedValue: 11,
        encodedValue: '0x0000000000000000000000000000000b',
      },
    ];

    validTestCases.forEach((testCase) => {
      it( encodes/decodes: ${testCase.decodedValue} as ${testCase.valueType} , () => {
        const encodedValue = encodeValueType(
          testCase.valueType,
          testCase.decodedValue,
        );

        assert.deepStrictEqual(encodedValue, testCase.encodedValue);
        assert.deepStrictEqual(
          decodeValueType(testCase.valueType, encodedValue),
          testCase.decodedValue,
        );
      });
    });
  });
```

```
describe('&amp;nbsp;uint256&amp;nbsp; type', () => {
  const validTestCases = [
    {
      valueType: 'uint256',
      decodedValue: 1337,
      encodedValue:
        '0x0000000000000000000000000000000000000000000000000000000000000539',
    },
  ];

  validTestCases.forEach((testCase) => {
    it(&amp;nbsp;encodes/decodes: ${testCase.decodedValue} as ${testCase.valueType}&amp;nbsp;, () => {
      const encodedValue = encodeValueType(
        testCase.valueType,
        testCase.decodedValue,
      );

      assert.deepStrictEqual(encodedValue, testCase.encodedValue);
      assert.deepStrictEqual(
        decodeValueType(testCase.valueType, encodedValue),
        testCase.decodedValue,
      );
    });
  });
});

describe('&amp;nbsp;string&amp;nbsp; type', () => {
  const validTestCases = [
    {
      valueType: 'string',
      decodedValue: 'Hello I am a string',
      encodedValue: '0x48656c6c6f204920616d20612073747269 6e67',
    },
  ];

  validTestCases.forEach((testCase) => {
    it(&amp;nbsp;encodes/decodes: ${testCase.decodedValue} as ${testCase.valueType}&amp;nbsp;, () => {
      const encodedValue = encodeValueType(
        testCase.valueType,
        testCase.decodedValue,
      );

      assert.deepStrictEqual(encodedValue, testCase.encodedValue);
      assert.deepStrictEqual(
        decodeValueType(testCase.valueType, encodedValue),
        testCase.decodedValue,
      );
    });
  });

  it('should encode each letter in a number as a utf8 character, and decode it back as a string', () => {
```

```
    const testCase = {
      valueType: 'string',
      decodedValue: 12345, // encode each letter as a utf8 hex
      encodedValue: '0x3132333435',
    };

    const encodedValue = encodeValueType(
      testCase.valueType,
      testCase.decodedValue,
    );

    assert.deepStrictEqual(encodedValue, testCase.encodedValue);
    assert.deepStrictEqual(
      decodeValueType(testCase.valueType, encodedValue),
       ${testCase.decodedValue} ,
    );
  });
});

describe(' address  type', () => {
  const validTestCases = [
    {
      valueType: 'address',
      // should decode as a checksummed address
      decodedValue: '0xdAC17F958D2ee523a2206206994597C13D831ec7',
      encodedValue: '0xdac17f958d2ee523a2206206994597c13d831ec7',
    },
  ];

  validTestCases.forEach((testCase) => {
    it( encodes/decodes: ${testCase.decodedValue} as ${testCase.valueType} , () => {
      const encodedValue = encodeValueType(
        testCase.valueType,
        testCase.decodedValue,
      );

      assert.deepStrictEqual(encodedValue, testCase.encodedValue);
      assert.deepStrictEqual(
        decodeValueType(testCase.valueType, encodedValue),
        testCase.decodedValue,
      );
    });
  });

  const errorEncodingTestCases = [
    {
      valueType: 'address',
      input: '0x388C818CA8B9251b3931', // less than 20 bytes
    },
    {
      valueType: 'address',
      input: '0x1f9090aaE28b8a3dCeaDf281B0F12828e676c326818CA8B92', // more than 20 bytes
```

```
      },
    ];

    errorEncodingTestCases.forEach((testCase) => {
      it( should throw when trying to encode ${testCase.input} as ${testCase.valueType} , async
() => {
        assert.throws(() =>
          encodeValueType(testCase.valueType, testCase.input),
        );
      });
    });
  });

  describe(' bytes  type', () => {
    const validTestCases = [
      {
        valueType: 'bytes',
        decodedValue:
          '0x0000000000000000000000000000000000000000000000000000000000001337',
        encodedValue:
          '0x0000000000000000000000000000000000000000000000000000000000001337',
      },
      {
        valueType: 'bytes',
        decodedValue: '0xaabbccddeeff1122334455',
        encodedValue: '0xaabbccddeeff1122334455',
      },
      {
        valueType: 'bytes',
        decodedValue: '0x1337',
        encodedValue: '0x1337',
      },
    ];

    validTestCases.forEach((testCase) => {
      it( encodes/decodes: ${testCase.decodedValue} as ${testCase.valueType} , () => {
        const encodedValue = encodeValueType(
          testCase.valueType,
          testCase.decodedValue,
        );

        assert.deepStrictEqual(encodedValue, testCase.encodedValue);
        assert.deepStrictEqual(
          decodeValueType(testCase.valueType, encodedValue),
          testCase.decodedValue,
        );
      });
    });
  });

  describe('arrays  []  of static types', () => {
    const validTestCases = [
```

```
    {
      valueType: 'string[]',
      decodedValue: ['a', 'b'],
      encodedValue:

'0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000
0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000
0000000000000000000400000000000000000000000000000000000000000000000000000000000000800
0000000000000000000000000000000000000000000000000000000000000161000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000001620000000000000000000000000000000000000000000000000000000000000000',
    },
    {
      valueType: 'address[]',
      decodedValue: [
        '0x68114e23B500Cdb63A5B6c9006f3acB0325AD0CC',
        '0x7466e40FEF4978394A07C9124ad4aD1A374b9465',
      ],
      encodedValue:

'0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000
0000000000000000000000000000000000000000000000000000000000000020000000000000000000000068114e23b500cdb63a
5b6c9006f3acb0325ad0cc0000000000000000000000007466e40fef4978394a07c9124ad4ad1a374b9465',
    },
    {
      valueType: 'uint256[]',
      decodedValue: [1, 99],
      encodedValue:

'0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000
0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000
0000000000000000000100000000000000000000000000000000000000000000000000000000000000000063'
,
    },
    {
      valueType: 'bytes32[]',
      decodedValue: [
        '0x1337000000000000000000000000000000000000000000000000000000000000',
        '0x0000000000000000000000000000000000000000000000000000000061626364',
      ],
      encodedValue:

'0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000
0000000000000000000000000000000000000000000000000000000000000021337000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000061626364'
,
    },
    {
      valueType: 'bytes4[]',
      decodedValue: ['0x12345678', '0x87654321'],
      encodedValue:
```

```
'0x000000000000000000000000000000000000000000000000000000000000002000000000000000000000
00000000000000000000000000000000000000000002123456780000000000000000000000000000000000
0000000000000000000000876543210000000000000000000000000000000000000000000000000000000000'
,
    },
    {
      valueType: 'bytes[]',
      decodedValue: [
        '0x0000000000000000000000000000000000000000000000000000000000001337',
        '0x00000000000000000000000000000000000000000000000000000000000054ef',
      ],
      encodedValue:

'0x000000000000000000000000000000000000000000000000000000000000002000000000000000000000
00000000000000000000000000000000000000000002000000000000000000000000000000000000000000
00000000000000000000400000000000000000000000000000000000000000000000000000000000000080
00000000000000000000000000000000000000000000000000000000000000200000000000000000000000
00000000000000000000000000000000000000133700000000000000000000000000000000000000000000
000000000000000000200000000000000000000000000000000000000000000000000000000000000054ef',
    },
    {
      valueType: 'bool[]',
      decodedValue: [true, false, true],
      encodedValue:

'0x000000000000000000000000000000000000000000000000000000000000002000000000000000000000
00000000000000000000000000000000000000000003000000000000000000000000000000000000000000
00000000000000000000010000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000001',
    },
    {
      valueType: 'bool[]',
      decodedValue: [false, false, true, false, false],
      encodedValue:

'0x000000000000000000000000000000000000000000000000000000000000002000000000000000000000
00000000000000000000000000000000000000000005000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000010000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000',
    },
    {
      valueType: 'boolean[]', // allow to specify "boolean"
      decodedValue: [true],
      encodedValue:

'0x000000000000000000000000000000000000000000000000000000000000002000000000000000000000
00000000000000000000000000000000000000000001000000000000000000000000000000000000000000
0000000000000000000001',
    },
    {
```

```
        valueType: 'boolean[]', // allow to specify "boolean"
        decodedValue: [false, false],
        encodedValue:

'0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000000'
,
      },
    ];

    validTestCases.forEach((testCase) => {
      it( encodes/decodes: ${testCase.decodedValue} as ${testCase.valueType} , () => {
        const encodedValue = encodeValueType(
          testCase.valueType,
          testCase.decodedValue,
        );

        assert.deepStrictEqual(encodedValue, testCase.encodedValue);
        assert.deepStrictEqual(
          decodeValueType(testCase.valueType, encodedValue),
          testCase.decodedValue,
        );
      });
    });
  });

  describe('when encoding a value that exceeds the maximal lenght of bytes than its type', () => {
    const validTestCases = [
      {
        valueType: 'bytes32',
        decodedValue:
          '0xcafecafecafecafecafecafecafecafecafecafecafecafecafecafecafecafecafecafe',
        encodedValue:
          '0xcafecafecafecafecafecafecafecafecafecafecafecafecafecafecafecafe',
      },
    ];

    validTestCases.forEach((testCase) => {
      it('should throw', async () => {
        assert.throws(() =>
          encodeValueType(testCase.valueType, testCase.decodedValue),
        );
      });
    });
  });

  describe('when encoding/decoding a value that is not a number as a  uint128 ', () => {
    it('throws when trying to encode a string as  uint128 ', () => {
      assert.throws(() => encodeValueType('uint128', 'helloWorld'));
    });
```

```
    it('throws when trying to encode a bytes17 as  uint128 ', () => {
      assert.throws(() =>
        encodeValueType('uint128', '340282366920938463463374607431768211456'),
      );
      assert.throws(() =>
        encodeValueType('uint128', '0x0100000000000000000000000000000000'),
      );
    });

    it('throws when trying to decode a bytes17 as  uint128 ', () => {
      expect(() =>
        decodeValueType('uint128', '0x00000000000000000000000000000ffff'),
      ).to.throw(
        "Can't convert hex value 0x00000000000000000000000000000ffff to uint128. Too many bytes. 17 >
16",
      );
    });
  });

  describe(' type[CompactBytesArray]  (of static types)', () => {
    const validTestCases = [
      {
        valueType: 'bytes[CompactBytesArray]',
        decodedValue: [
          '0xaabb',
          '0xcafecafecafecafecafecafecafecafecafecafecafecafecafecafecafecafe',
          '0xbeefbeefbeefbeefbeef',
        ],
        encodedValue:

'0x0002aabb0020cafecafecafecafecafecafecafecafecafecafecafecafecafecafecafecafe000abeefbeefbeefbee
fbeef',
      },
      {
        valueType: 'bytes[CompactBytesArray]',
        decodedValue: [ 0x${'cafe'.repeat(256)} ,  0x${'beef'.repeat(250)} ],
        encodedValue:  0x0200${'cafe'.repeat(256)}01f4${'beef'.repeat(250)} ,
      },
      {
        valueType: 'string[CompactBytesArray]',
        decodedValue: [
          'one random string',
          'bring back my coke',
          'Diagon Alley',
        ],
        encodedValue:  0x0011${stripHexPrefix(
          utf8ToHex('one random string'),
        )}0012${stripHexPrefix(
          utf8ToHex('bring back my coke'),
        )}000c${stripHexPrefix(utf8ToHex('Diagon Alley'))} ,
      },
      {
```

```
    valueType: 'uint8[CompactBytesArray]',
    decodedValue: [1, 43, 73, 255],
    encodedValue: '0x00010100012b0001490001ff',
  },
  {
    valueType: 'bytes4[CompactBytesArray]',
    decodedValue: [
      '0xe6520726',
      '0x272696e6',
      '0x72062616',
      '0xab7f11e3',
    ],
    encodedValue: '0x0004e65207260004272696e60004720626160004ab7f11e3',
  },
];

validTestCases.forEach((testCase) => {
  it( encodes/decodes: ${testCase.decodedValue} as ${testCase.valueType} , () => {
    const encodedValue = encodeValueType(
      testCase.valueType,
      testCase.decodedValue,
    );

    assert.deepStrictEqual(encodedValue, testCase.encodedValue);
    assert.deepStrictEqual(
      decodeValueType(testCase.valueType, encodedValue),
      testCase.decodedValue,
    );
  });
});

describe('when encoding bytes[CompactBytesArray]', () => {
  it('should encode  0x  elements as  0x0000 ', async () => {
    const testCase = {
      valueType: 'bytes[CompactBytesArray]',
      decodedValue: ['0xaabb', '0x', '0x', '0xbeefbeefbeefbeefbeef'],
      encodedValue: '0x0002aabb000000000000abeefbeefbeefbeefbeef',
    };

    const encodedValue = encodeValueType(
      testCase.valueType,
      testCase.decodedValue,
    );
    assert.deepStrictEqual(encodedValue, testCase.encodedValue);
  });

  it("should encode '' (empty strings) elements as  0x0000 ", async () => {
    const testCase = {
      valueType: 'bytes[CompactBytesArray]',
      decodedValue: ['0xaabb', '', '', '0xbeefbeefbeefbeefbeef'],
      encodedValue: '0x0002aabb000000000000abeefbeefbeefbeefbeef',
    };
```

```
        const encodedValue = encodeValueType(
          testCase.valueType,
          testCase.decodedValue,
        );
        assert.deepStrictEqual(encodedValue, testCase.encodedValue);
      });

      it('should throw when trying to encode a array that contains non hex string as
&amp;nbsp;bytes[CompactBytesArray]&amp;nbsp;', async () => {
        expect(() => {
          encodeValueType('bytes[CompactBytesArray]', [
            'some random string',
            'another random strings',
            '0xaabbccdd',
          ]);
        }).to.throw(
          "Couldn't encode bytes[CompactBytesArray], value at index 0 is not hex",
        );
      });

      it('should throw when trying to encode a &amp;nbsp;bytes[CompactBytesArray]&amp;nbsp; with a bytes length
bigger than 65_535', async () => {
        expect(() => {
          encodeValueType('bytes[CompactBytesArray]', [
            '0x' + 'ab'.repeat(66_0000),
          ]);
        }).to.throw(
          "Couldn't encode bytes[CompactBytesArray], value at index 0 exceeds 65_535 bytes",
        );
      });
    });

    describe('when encoding uintN[CompactBytesArray]', () => {
      it('should throw if trying to encode a value that exceeds the maximal lenght of bytes for this type', async
() => {
        expect(() => {
          encodeValueType('uint8[CompactBytesArray]', [15, 178, 266]);
        }).to.throw('Hex uint8 value at index 2 does not fit in 1 bytes');
      });

      it('should throw if trying to decode a value that exceeds the maximal lenght of bytes for this type', async
() => {
        expect(() => {
          decodeValueType(
            'uint8[CompactBytesArray]',
            '0x00010100012b00014900020100',
          );
        }).to.throw('Hex uint8 value at index 3 does not fit in 1 bytes');
      });
    });
```

```
    describe('when encoding bytesN[CompactBytesArray]', () => {
      it('should throw if trying to encode a value that exceeds the maximal lenght of bytes for this type', async
() => {
        expect(() => {
          encodeValueType('bytes4[CompactBytesArray]', [
            '0xe6520726',
            '0x272696e6',
            '0x72062616',
            '0xab7f11e3aabbcc',
          ]);
        }).to.throw('Hex bytes4 value at index 3 does not fit in 4 bytes');
      });

      it('should throw if trying to decode a value that exceeds the maximal lenght of bytes for this type', async
() => {
        expect(() => {
          decodeValueType(
            'bytes4[CompactBytesArray]',
            '0x0004e65207260004272696e60004720626160007ab7f11e3aabbcc',
          );
        }).to.throw('Hex bytes4 value at index 3 does not fit in 4 bytes');
      });
    });

    describe('when decoding a bytes[CompactBytesArray] that contains &amp;nbsp;0000&amp;nbsp; entries', () => {
      it("should decode as '' (empty string) in the decoded array", async () => {
        const testCase = {
          valueType: 'bytes[CompactBytesArray]',
          decodedValue: ['0xaabb', '', '', '0xbeefbeefbeefbeefbeef'],
          encodedValue: '0x0002aabb00000000000abeefbeefbeefbeefbeef',
        };

        const decodedValue = decodeValueType(
          testCase.valueType,
          testCase.encodedValue,
        );
        assert.deepStrictEqual(decodedValue, testCase.decodedValue);
      });

      it('should throw when trying to decode a &amp;nbsp;bytes[CompactBytesArray]&amp;nbsp; with an invalid length
byte', async () => {
        expect(() => {
          decodeValueType('bytes[CompactBytesArray]', '0x0005cafe');
        }).to.throw("Couldn't decode bytes[CompactBytesArray]");
      });
    });
  });

  it('should throw when valueType is unknown', () => {
    assert.throws(() => {
      encodeValueType('????', 'hi');
    });
```

```
    assert.throws(() => {
      decodeValueType('whatIsthisType', 'hi');
    });
  });
});

describe('valueContent', () => {
  const testCases = [
    {
      valueContent: 'Keccak256',
      decodedValue:
        '7f37518252ad8c46b3eecd357685e7cd0e2ed88534c10751b1b81ac04dc40bc3',
      encodedValue:
        '7f37518252ad8c46b3eecd357685e7cd0e2ed88534c10751b1b81ac04dc40bc3',
    },
    {
      valueContent: 'Number',
      decodedValue: 876,
      encodedValue:
        '0x000000000000000000000000000000000000000000000000000000000000036c',
    },
    {
      valueContent: 'Address',
      decodedValue: '0xa29Afb8F3ccE086B3992621324E9d7c104F03D1B',
      encodedValue: '0xa29afb8f3cce086b3992621324e9d7c104f03d1b',
    },
    {
      valueContent: 'String',
      decodedValue: 'hello',
      encodedValue: '0x68656c6c6f',
    },
    {
      valueContent: 'Markdown',
      decodedValue: '# hello',
      encodedValue: '0x2320686 56c6c6f',
    },
    {
      valueContent: 'URL',
      decodedValue: 'http://test.com',
      encodedValue: '0x687474703a2f2f746573742e636f6d',
    },
    {
      valueContent: 'AssetURL',
      decodedValue: {
        verification: {
          method: SUPPORTED_VERIFICATION_METHOD_STRINGS.KECCAK256_UTF8,
          data: '0x027547537d35728a741470df1ccf65de10b454ca0def7c5c20b257b7b8d16168',
        },
        url: 'http://test.com/asset.glb',
      },
      encodedValue:
```

```
'0x6f357c6a027547537d35728a741470df1ccf65de10b454ca0def7c5c20b257b7b8d16168687474703a2f2f746573742e636f6d2f61737365742e676c62',
    },
    {
      valueContent: 'BitArray',
      encodedValue:
        '0x0000000000000000000000000000000000000000000000000000000000000008', // ... 0000 0000 1000
      decodedValue:
        '0x0000000000000000000000000000000000000000000000000000000000000008',
    },
    {
      valueContent: 'Boolean',
      encodedValue: '0x00',
      decodedValue: false,
    },
    {
      valueContent: 'Boolean',
      encodedValue: '0x01',
      decodedValue: true,
    },
  ];

  testCases.forEach((testCase) => {
    it(&amp;nbsp;encodes/decodes: ${testCase.valueContent}&amp;nbsp;, () => {
      const encodedValue = encodeValueContent(
        testCase.valueContent,
        testCase.decodedValue,
      );

      encodeValueContent(testCase.valueContent, testCase.decodedValue);

      assert.deepStrictEqual(encodedValue, testCase.encodedValue);
      assert.deepStrictEqual(
        encodedValue !== false
          ? decodeValueContent(testCase.valueContent, encodedValue)
          : false,
        testCase.decodedValue,
      );
    });
  });

  it('encodes/decodes: JSONURL', () => {
    const dataToEncode: JSONURLDataToEncode = {
      url: 'ipfs://QmYr1VJLwerg6pEoscdhVGugo39pa6rycEZLjtRPDfW84UAx',
      json: {
        myProperty: 'is a string',
        anotherProperty: {
          key: 123456,
        },
      },
    };
```

```
    const encodedValue = encodeValueContent('JSONURL', dataToEncode);

    const verificationMethod =
      SUPPORTED_VERIFICATION_METHOD_HASHES.HASH_KECCAK256_UTF8;
    const hexUrl = utf8ToHex(dataToEncode.url).substring(2);
    const jsonVerificationData = keccak256(
      JSON.stringify(dataToEncode.json),
    ).substring(2);
    assert.deepStrictEqual(
      encodedValue,
      verificationMethod + jsonVerificationData + hexUrl,
    );

    const expectedDecodedValue: URLDataWithHash = {
      verification: {
        data:  0x${jsonVerificationData} ,
        method: SUPPORTED_VERIFICATION_METHOD_STRINGS.KECCAK256_UTF8,
      },
      url: dataToEncode.url,
    };

    assert.deepStrictEqual(
      encodedValue !== false
        ? decodeValueContent('JSONURL', encodedValue)
        : false,
      expectedDecodedValue,
    );
  });

  it('should throw when encodeValueContent value is a string and valueContent is JSONURL or
ASSETURL', () => {
    expect(() => {
      encodeValueContent('AssetURL', 'imnotanobject!');
    }).to.throw;
    expect(() => {
      encodeValueContent('JSONURL', 'imnotanobject!');
    }).to.throw;
  });

  it('should throw when valueContent is unknown', () => {
    assert.throws(() => {
      encodeValueContent('wrongContent', 'hi');
    });
    assert.throws(() => {
      decodeValueContent('wrongContent', 'hi');
    });
  });

  it('should return the value if the valueContent == value', () => {
    const value =
      '0x027547537d35728a741470df1ccf65de10b454ca0def7c5c20b257b7b8d16168';
```

```
    assert.deepStrictEqual(encodeValueContent(value, value), value);
    assert.deepStrictEqual(decodeValueContent(value, value), value);
  });

  describe('JSONURL', () => {
    it('throws when the verification method of JSON of JSONURL to encode is not keccak256(utf8)', () => {
      assert.throws(() => {
        valueContentEncodingMap('JSONURL').encode({
          // @ts-ignore to still run the test (incase someone is using the library in a non TS environment)
          verification: {
            method: 'whatever',
            data: '0x321',
          },
          url: 'https://file-desination.com/file-name',
        });
      }, &amp;nbsp;Chosen verification method 'whatever' is not supported. Supported verification methods:
keccak256(utf8),keccak256(bytes)&amp;nbsp;);
    });

    it('throws when JSONURL encode a JSON without json or verificationData key', () => {
      assert.throws(() => {
        valueContentEncodingMap('JSONURL').encode({
          // @ts-ignore to still run the test (incase someone is using the library in a non TS environment)
          verification: {
            method: 'keccak256(utf8)',
          },
          url: 'https://file-desination.com/file-name',
        });
      }, 'You have to provide either the verification.data or the json via the respective properties');
    });
  });
 });
});
```

</file>

<file>

path: /src/lib/encoder.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/lib/encoder.ts

```
/*
  This file is part of @erc725/erc725.js.
  @erc725/erc725.js is free software: you can redistribute it and/or modify
  it under the terms of the GNU Lesser General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.
  @erc725/erc725.js is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
```

```
    GNU Lesser General Public License for more details.
    You should have received a copy of the GNU Lesser General Public License
    along with @erc725/erc725.js.  If not, see <http://www.gnu.org/licenses/>.
*/
/**
 * @file lib/encoder.ts
 * @author Robert McLeod <@robertdavid010>
 * @author Fabian Vogelsteller <fabian@lukso.network>
 * @author Hugo Masclet <@Hugoo>
 * @author Callum Grindle <@CallumGrindle>
 * @author Jean Cavallera <@CJ42>
 * @date 2023
 */

/*
  this handles encoding and decoding as per necessary for the erc725 schema specifications
*/

import AbiCoder from 'web3-eth-abi';

import {
  hexToNumber,
  hexToUtf8,
  isAddress,
  isHex,
  keccak256,
  numberToHex,
  padLeft,
  toChecksumAddress,
  utf8ToHex,
  stripHexPrefix,
  hexToBytes,
  bytesToHex,
  toHex,
  toBN,
} from 'web3-utils';

import { JSONURLDataToEncode, URLDataWithHash, Verification } from '../types';
import { AssetURLEncode } from '../types/encodeData';

import {
  SUPPORTED_VERIFICATION_METHOD_STRINGS,
  UNKNOWN_VERIFICATION_METHOD,
} from '../constants/constants';
import { getVerificationMethod, hashData, countNumberOfBytes } from './utils';

const abiCoder = AbiCoder;

const bytesNRegex = /Bytes(\d+)/;

const ALLOWED_BYTES_SIZES = [2, 4, 8, 16, 32, 64, 128, 256];
```

```
const encodeDataSourceWithHash = (
  verification: undefined | Verification,
  dataSource: string,
): string => {
  const verificationMethod = getVerificationMethod(
    verification?.method || UNKNOWN_VERIFICATION_METHOD,
  );
  return (
    (verificationMethod
      ? keccak256(verificationMethod.name).slice(0, 10)
      : padLeft(0, 8)) +
    stripHexPrefix(verification ? verification.data : padLeft(0, 64)) +
    stripHexPrefix(utf8ToHex(dataSource))
  );
};

const decodeDataSourceWithHash = (value: string): URLDataWithHash => {
  const verificationMethodSig = value.slice(0, 10);
  const verificationMethod = getVerificationMethod(verificationMethodSig);

  const encodedData = value.replace('0x', '').slice(8); // Rest of data string after function hash
  const dataHash = '0x' + encodedData.slice(0, 64); // Get jsonHash 32 bytes
  const dataSource = hexToUtf8('0x' + encodedData.slice(64)); // Get remainder as URI

  return {
    verification: {
      method: verificationMethod?.name || UNKNOWN_VERIFICATION_METHOD,
      data: dataHash,
    },
    url: dataSource,
  };
};

const encodeToBytesN = (
  bytesN: 'bytes32' | 'bytes4',
  value: string | number,
): string => {
  let valueToEncode: string;

  if (typeof value === 'string' && !isHex(value)) {
    // if we receive a plain string (e.g: "hey!"), convert it to utf8-hex data
    valueToEncode = toHex(value);
  } else if (typeof value === 'number') {
    // if we receive a number as input, convert it to hex
    valueToEncode = numberToHex(value);
  } else {
    valueToEncode = value;
  }

  const numberOfBytesInType = parseInt(bytesN.split('bytes')[1], 10);
  const numberOfBytesInValue = countNumberOfBytes(valueToEncode);
```

```
  if (numberOfBytesInValue > numberOfBytesInType) {
    throw new Error(
       Can't convert ${value} to ${bytesN}. Too many bytes, expected at most ${numberOfBytesInType}
bytes, received ${numberOfBytesInValue}. ,
    );
  }

  const abiEncodedValue = abiCoder.encodeParameter(bytesN, valueToEncode);

  // abi-encoding right pads to 32 bytes, if we need less, we need to remove the padding
  if (numberOfBytesInType === 32) {
    return abiEncodedValue;
  }

  const bytesArray = hexToBytes(abiEncodedValue);
  return bytesToHex(bytesArray.slice(0, 4));
};

/**
 * Encodes bytes to CompactBytesArray
 *
 * @param values An array of BytesLike strings
 * @returns bytes[CompactBytesArray]
 */
const encodeCompactBytesArray = (values: string[]): string => {
  const compactBytesArray = values
    .filter((value, index) => {
      if (!isHex(value)) {
        throw new Error(
           Couldn't encode bytes[CompactBytesArray], value at index ${index} is not hex ,
        );
      }

      if (value.length > 65_535 * 2 + 2) {
        throw new Error(
           Couldn't encode bytes[CompactBytesArray], value at index ${index} exceeds 65_535
bytes ,
        );
      }

      return true;
    })
    .reduce((acc, value) => {
      const numberOfBytes = stripHexPrefix(value).length / 2;
      const hexNumber = padLeft(numberToHex(numberOfBytes), 4);
      return acc + stripHexPrefix(hexNumber) + stripHexPrefix(value);
    }, '0x');

  return compactBytesArray;
};

/**
```

```
 * Decodes CompactBytesArray of type bytes
 *
 * @param compactBytesArray A bytes[CompactBytesArray]
 * @returns An array of BytesLike strings decode from &amp;nbsp;compactBytesArray&amp;nbsp;
 */
const decodeCompactBytesArray = (compactBytesArray: string): string[] => {
  if (!isHex(compactBytesArray))
    throw new Error("Couldn't decode, value is not hex");

  let pointer = 0;
  const encodedValues: string[] = [];

  const strippedCompactBytesArray = stripHexPrefix(compactBytesArray);

  while (pointer < strippedCompactBytesArray.length) {
    const length = Number(
      hexToNumber('0x' + strippedCompactBytesArray.slice(pointer, pointer + 4)),
    );

    if (length === 0) {
      // empty entries (&amp;nbsp;0x0000&amp;nbsp;) in a CompactBytesArray are returned as empty entries in the
array
      encodedValues.push('');
    } else {
      encodedValues.push(
        '0x' +
          strippedCompactBytesArray.slice(
            pointer + 4,
            pointer + 2 * (length + 2),
          ),
      );
    }

    pointer += 2 * (length + 2);
  }

  if (pointer > strippedCompactBytesArray.length)
    throw new Error("Couldn't decode bytes[CompactBytesArray]");

  return encodedValues;
};

/**
 * Encodes bytesN to CompactBytesArray
 *
 * @param values An array of BytesLike strings
 * @param numberOfBytes The number of bytes for each value from &amp;nbsp;values&amp;nbsp;
 * @returns bytesN[CompactBytesArray]
 */
const encodeBytesNCompactBytesArray = (
  values: string[],
  numberOfBytes: number,
```

```
): string => {
  values.forEach((value, index) => {
    if (stripHexPrefix(value).length > numberOfBytes * 2)
      throw new Error(
         Hex bytes${numberOfBytes} value at index ${index} does not fit in ${numberOfBytes}
bytes ,
      );
  });

  return encodeCompactBytesArray(values);
};

/**
 * Decodes CompactBytesArray of type bytesN
 *
 * @param compactBytesArray A bytesN[CompactBytesArray]
 * @param numberOfBytes The number of bytes allowed per each element from
 compactBytesArray 
 * @returns An array of BytesLike strings decoded from  compactBytesArray 
 */
const decodeBytesNCompactBytesArray = (
  compactBytesArray: string,
  numberOfBytes: number,
): string[] => {
  const bytesValues = decodeCompactBytesArray(compactBytesArray);
  bytesValues.forEach((bytesValue, index) => {
    if (stripHexPrefix(bytesValue).length > numberOfBytes * 2)
      throw new Error(
         Hex bytes${numberOfBytes} value at index ${index} does not fit in ${numberOfBytes}
bytes ,
      );
  });

  return bytesValues;
};

/**
 * @returns Encoding/decoding for bytes1[CompactBytesArray] to bytes32[COmpactBytesArray]
 */
const returnTypesOfBytesNCompactBytesArray = () => {
  const types: Record<
    string,
    { encode: (value: string[]) => string; decode: (value: string) => string[] }
  > = {};

  for (let i = 1; i < 33; i++) {
    types[ bytes${i}[CompactBytesArray] ] = {
      encode: (value: string[]) => encodeBytesNCompactBytesArray(value, i),
      decode: (value: string) => decodeBytesNCompactBytesArray(value, i),
    };
  }
  return types;
```

```
};

/**
 * Encodes uintN to CompactBytesArray
 * @param values An array of BytesLike strings
 * @param numberOfBytes The number of bytes for each value from  values 
 * @returns uintN[CompactBytesArray]
 */
const encodeUintNCompactBytesArray = (
  values: number[],
  numberOfBytes: number,
): string => {
  const hexValues: string[] = values.map((value, index) => {
    const hexNumber = stripHexPrefix(numberToHex(value)).padStart(
      numberOfBytes * 2,
      '0',
    );
    if (hexNumber.length > numberOfBytes * 2)
      throw new Error(
         Hex uint${
          numberOfBytes * 8
        } value at index ${index} does not fit in ${numberOfBytes} bytes ,
      );
    return hexNumber;
  });

  return encodeCompactBytesArray(hexValues);
};

/**
 * Decodes CompactBytesArray of type uintN
 * @param compactBytesArray A uintN[CompactBytesArray]
 * @param numberOfBytes The number of bytes allowed per each element from
 compactBytesArray 
 * @returns An array of numbers decoded from  compactBytesArray 
 */
const decodeUintNCompactBytesArray = (
  compactBytesArray: string,
  numberOfBytes: number,
): number[] => {
  const hexValues = decodeCompactBytesArray(compactBytesArray);

  return hexValues.map((hexValue, index) => {
    const hexValueStripped = stripHexPrefix(hexValue);
    if (hexValueStripped.length > numberOfBytes * 2)
      throw new Error(
         Hex uint${
          numberOfBytes * 8
        } value at index ${index} does not fit in ${numberOfBytes} bytes ,
      );
    return Number(hexToNumber(hexValue));
  });
```

```
};

/**
 * @returns Encoding/decoding for uint8[CompactBytesArray] to uint256[COmpactBytesArray]
 */
const returnTypesOfUintNCompactBytesArray = () => {
  const types: Record<
    string,
    { encode: (value: number[]) => string; decode: (value: string) => number[] }
  > = {};

  for (let i = 1; i < 33; i++) {
    types[&amp;nbsp;uint${i * 8}[CompactBytesArray]&amp;nbsp;] = {
      encode: (value: number[]) => encodeUintNCompactBytesArray(value, i),
      decode: (value: string) => decodeUintNCompactBytesArray(value, i),
    };
  }
  return types;
};

/**
 * Encodes any set of strings to string[CompactBytesArray]
 *
 * @param values An array of non restricted strings
 * @returns string[CompactBytesArray]
 */
const encodeStringCompactBytesArray = (values: string[]): string => {
  const hexValues: string[] = values.map((element) => utf8ToHex(element));

  return encodeCompactBytesArray(hexValues);
};

/**
 * Decode a string[CompactBytesArray] to an array of strings
 * @param compactBytesArray A string[CompactBytesArray]
 * @returns An array of strings
 */
const decodeStringCompactBytesArray = (compactBytesArray: string): string[] => {
  const hexValues: string[] = decodeCompactBytesArray(compactBytesArray);
  const stringValues: string[] = hexValues.map((element) => hexToUtf8(element));

  return stringValues;
};

const valueTypeEncodingMap = {
  bool: {
    encode: (value: boolean) => (value ? '0x01' : '0x00'),
    decode: (value: string) => value === '0x01',
  },
  boolean: {
    encode: (value: boolean) => (value ? '0x01' : '0x00'),
    decode: (value: string) => value === '0x01',
```

```
},
string: {
  encode: (value: string | number) => {
    // if we receive a number as input,
    // convert each letter to its utf8 hex representation
    if (typeof value === 'number') {
      return utf8ToHex(&amp;nbsp;${value}&amp;nbsp;);
    }

    return utf8ToHex(value);
  },
  decode: (value: string) => hexToUtf8(value),
},
address: {
  encode: (value: string) => {
    // abi-encode pads to 32 x 00 bytes on the left, so we need to remove them
    const abiEncodedValue = abiCoder.encodeParameter('address', value);

    // convert to an array of individual bytes
    const bytesArray = hexToBytes(abiEncodedValue);

    // just keep the last 20 bytes, starting at index 12
    return bytesToHex(bytesArray.slice(12));
  },
  decode: (value: string) => toChecksumAddress(value),
},
// NOTE: We could add conditional handling of numeric values here...
uint128: {
  encode: (value: string | number) => {
    const abiEncodedValue = abiCoder.encodeParameter('uint128', value);
    const bytesArray = hexToBytes(abiEncodedValue);
    return bytesToHex(bytesArray.slice(16));
  },
  decode: (value: string) => {
    if (!isHex(value)) {
      throw new Error(&amp;nbsp;Can't convert ${value} to uint128, value is not hex.&amp;nbsp;);
    }

    if (value.length > 34) {
      throw new Error(
        &amp;nbsp;Can't convert hex value ${value} to uint128. Too many bytes. ${
          (value.length - 2) / 2
        } > 16&amp;nbsp;,
      );
    }

    return toBN(value).toNumber();
  },
},
uint256: {
  encode: (value: string | number) => {
    return abiCoder.encodeParameter('uint256', value);
```

```
  },
  decode: (value: string) => {
    if (!isHex(value)) {
      throw new Error(&amp;nbsp;Can't convert ${value} to uint256, value is not hex.&amp;nbsp;);
    }

    const numberOfBytes = countNumberOfBytes(value);

    if (numberOfBytes > 32) {
      throw new Error(
        &amp;nbsp;Can't convert hex value ${value} to uint256. Too many bytes. ${numberOfBytes} is above the
maximal number of bytes 32.&amp;nbsp;,
      );
    }

    return toBN(value).toNumber();
  },
},
bytes32: {
  encode: (value: string | number) => encodeToBytesN('bytes32', value),
  decode: (value: string) => abiCoder.decodeParameter('bytes32', value),
},
bytes4: {
  encode: (value: string | number) => encodeToBytesN('bytes4', value),
  decode: (value: string) => {
    // we need to abi-encode the value again to ensure that:
    //  - that data to decode does not go over 4 bytes.
    //  - if the data is less than 4 bytes, that it gets padded to 4 bytes long.
    const reEncodedData = abiCoder.encodeParameter('bytes4', value);
    return abiCoder.decodeParameter('bytes4', reEncodedData);
  },
},
bytes: {
  encode: (value: string) => toHex(value),
  decode: (value: string) => value,
},
'bool[]': {
  encode: (value: boolean) => abiCoder.encodeParameter('bool[]', value),
  decode: (value: string) => abiCoder.decodeParameter('bool[]', value),
},
'boolean[]': {
  encode: (value: boolean) => abiCoder.encodeParameter('bool[]', value),
  decode: (value: string) => abiCoder.decodeParameter('bool[]', value),
},
'string[]': {
  encode: (value: string[]) => abiCoder.encodeParameter('string[]', value),
  decode: (value: string) => abiCoder.decodeParameter('string[]', value),
},
'address[]': {
  encode: (value: string[]) => abiCoder.encodeParameter('address[]', value),
  decode: (value: string) => abiCoder.decodeParameter('address[]', value),
},
```

```
  'uint256[]': {
    encode: (value: Array<number | string>) =>
      abiCoder.encodeParameter('uint256[]', value),
    decode: (value: string) => {
      // we want to return an array of numbers as [1, 2, 3], not an array of strings as [ '1', '2', '3']
      return abiCoder
        .decodeParameter('uint256[]', value)
        .map((numberAsString) => parseInt(numberAsString, 10));
    },
  },
  'bytes32[]': {
    encode: (value: string[]) => abiCoder.encodeParameter('bytes32[]', value),
    decode: (value: string) => abiCoder.decodeParameter('bytes32[]', value),
  },
  'bytes4[]': {
    encode: (value: string[]) => abiCoder.encodeParameter('bytes4[]', value),
    decode: (value: string) => abiCoder.decodeParameter('bytes4[]', value),
  },
  'bytes[]': {
    encode: (value: string[]) => abiCoder.encodeParameter('bytes[]', value),
    decode: (value: string) => abiCoder.decodeParameter('bytes[]', value),
  },
  'bytes[CompactBytesArray]': {
    encode: (value: string[]) => encodeCompactBytesArray(value),
    decode: (value: string) => decodeCompactBytesArray(value),
  },
  'string[CompactBytesArray]': {
    encode: (value: string[]) => encodeStringCompactBytesArray(value),
    decode: (value: string) => decodeStringCompactBytesArray(value),
  },
  ...returnTypesOfBytesNCompactBytesArray(),
  ...returnTypesOfUintNCompactBytesArray(),
};

// Use enum for type below
// Is it this enum ERC725JSONSchemaValueType? (If so, custom is missing from enum)

export const valueContentEncodingMap = (valueContent: string) => {
  const bytesNRegexMatch = valueContent.match(bytesNRegex);
  const bytesLength = bytesNRegexMatch ? parseInt(bytesNRegexMatch[1], 10) : '';

  switch (valueContent) {
    case 'Keccak256': {
      return {
        type: 'bytes32',
        encode: (value: string) => value,
        decode: (value: string) => value,
      };
    }
    case 'Number': {
      return {
        type: 'uint256',
```

```
    encode: (value: string) => {
      let parsedValue: number;
      try {
        parsedValue = parseInt(value, 10);
      } catch (error: any) {
        throw new Error(error);
      }

      return padLeft(numberToHex(parsedValue), 64);
    },
    decode: (value) => Number(hexToNumber(value)),
  };
}
// NOTE: This is not symmetrical, and always returns a checksummed address
case 'Address': {
  return {
    type: 'address',
    encode: (value: string) => {
      if (isAddress(value)) {
        return value.toLowerCase();
      }

      throw new Error('Address: "' + value + '" is an invalid address.');
    },
    decode: (value: string) => toChecksumAddress(value),
  };
}
case 'String': {
  return {
    type: 'string',
    encode: (value: string) => utf8ToHex(value),
    decode: (value: string) => hexToUtf8(value),
  };
}
case 'Markdown': {
  return {
    type: 'string',
    encode: (value: string) => utf8ToHex(value),
    decode: (value: string) => hexToUtf8(value),
  };
}
case 'URL': {
  return {
    type: 'string',
    encode: (value: string) => utf8ToHex(value),
    decode: (value: string) => hexToUtf8(value),
  };
}
case 'AssetURL': {
  return {
    type: 'custom',
    encode: (value: AssetURLEncode) =>
```

```
        encodeDataSourceWithHash(value.verification, value.url),
      decode: (value: string) => decodeDataSourceWithHash(value),
    };
  }
  // https://github.com/lukso-network/LIPs/blob/master/LSPs/LSP-2-ERC725YJSONSchema.md#jsonurl
  case 'JSONURL': {
    return {
      type: 'custom',
      encode: (dataToEncode: JSONURLDataToEncode) => {
        const {
          verification: { data, method } = {},
          json,
          url,
        } = dataToEncode;

        let hashedJson = data;

        if (json) {
          if (method) {
            throw new Error(
              'When passing in the  json  property, we use "keccak256(utf8)" as a default
verification method. You do not need to set a  verification.method .',
            );
          }
          hashedJson = hashData(
            json,
            SUPPORTED_VERIFICATION_METHOD_STRINGS.KECCAK256_UTF8,
          );
        }

        if (!hashedJson) {
          throw new Error(
            'You have to provide either the verification.data or the json via the respective properties',
          );
        }

        return encodeDataSourceWithHash(
          {
            method:
              (method as SUPPORTED_VERIFICATION_METHOD_STRINGS) ||
              SUPPORTED_VERIFICATION_METHOD_STRINGS.KECCAK256_UTF8,
            data: hashedJson,
          },
          url,
        );
      },
      decode: (dataToDecode: string) =>
        decodeDataSourceWithHash(dataToDecode),
    };
  }
  case  Bytes${bytesLength} : {
    return {
```

```
      type: 'bytes',
      encode: (value: string) => {
        if (typeof value !== 'string' || !isHex(value)) {
          throw new Error( Value: ${value} is not hex. );
        }

        if (bytesLength && !ALLOWED_BYTES_SIZES.includes(bytesLength)) {
          throw new Error(
             Provided bytes length: ${bytesLength} for encoding valueContent: ${valueContent} is not
valid. ,
          );
        }

        if (bytesLength && value.length !== 2 + bytesLength * 2) {
          throw new Error(
             Value: ${value} is not of type ${valueContent}. Expected hex value of length ${
              2 + bytesLength * 2
            } ,
          );
        }

        return value;
      },
      decode: (value: string) => {
        if (typeof value !== 'string' || !isHex(value)) {
          console.log( Value: ${value} is not hex. );
          return null;
        }

        if (bytesLength && !ALLOWED_BYTES_SIZES.includes(bytesLength)) {
          console.error(
             Provided bytes length: ${bytesLength} for encoding valueContent: ${valueContent} is not
valid. ,
          );
          return null;
        }

        if (bytesLength && value.length !== 2 + bytesLength * 2) {
          console.error(
             Value: ${value} is not of type ${valueContent}. Expected hex value of length ${
              2 + bytesLength * 2
            } ,
          );
          return null;
        }

        return value;
      },
    };
  }
  case 'BitArray': {
    return {
```

```
      type: 'bytes',
      encode: (value: string) => {
        if (typeof value !== 'string' || !isHex(value)) {
          throw new Error( Value: ${value} is not hex. );
        }

        return value;
      },
      decode: (value: string) => {
        if (typeof value !== 'string' || !isHex(value)) {
          console.error( Value: ${value} is not hex. );
          return null;
        }

        return value;
      },
    };
  }
  case 'Boolean': {
    return {
      type: 'bool',
      encode: (value): string => {
        return valueTypeEncodingMap.bool.encode(value);
      },
      decode: (value: string): boolean => {
        try {
          return valueTypeEncodingMap.bool.decode(value) as any as boolean;
        } catch (error) {
          throw new Error( Value ${value} is not a boolean );
        }
      },
    };
  }
  default: {
    return {
      type: 'unknown',
      // eslint-disable-next-line @typescript-eslint/no-unused-vars
      encode: (_value: any) => {
        throw new Error(
           Could not encode unknown (${valueContent}) valueContent. ,
        );
      },
      // eslint-disable-next-line @typescript-eslint/no-unused-vars
      decode: (_value: any) => {
        throw new Error(
           Could not decode unknown (${valueContent}) valueContent. ,
        );
      },
    };
  }
 }
};
```

```
export function encodeValueType(
  type: string,
  value: string | string[] | number | number[] | boolean | boolean[],
): string {
  if (!valueTypeEncodingMap[type]) {
    throw new Error('Could not encode valueType: "' + type + '".');
  }

  if (typeof value === 'undefined' || value === null) {
    return value;
  }

  return valueTypeEncodingMap[type].encode(value);
}

export function decodeValueType(type: string, value: string) {
  if (!valueTypeEncodingMap[type]) {
    throw new Error('Could not decode valueType: "' + type + '".');
  }

  if (value === '0x') return null;

  if (typeof value === 'undefined' || value === null) {
    return value;
  }

  return valueTypeEncodingMap[type].decode(value);
}

export function encodeValueContent(
  valueContent: string,
  value: string | number | AssetURLEncode | JSONURLDataToEncode | boolean,
): string | false {
  if (valueContent.slice(0, 2) === '0x') {
    return valueContent === value ? value : false;
  }

  const valueContentEncodingMethods = valueContentEncodingMap(valueContent);

  if (!valueContentEncodingMethods) {
    throw new Error( Could not encode valueContent: ${valueContent}. );
  }

  if (value === null || value === undefined) {
    return '0x';
  }

  if (
    (valueContent === 'AssetURL' ||
      valueContent === 'JSONURL' ||
      valueContent === 'Boolean') &&
```

```
    typeof value === 'string'
  ) {
    const expectedValueType = valueContent === 'Boolean' ? 'boolean' : 'object';

    throw new Error(
       Could not encode valueContent: ${valueContent} with value: ${value}. Expected
${expectedValueType}. ,
    );
  }

  return valueContentEncodingMethods.encode(value as any) as string;
}

export function decodeValueContent(
  valueContent: string,
  value: string,
): string | URLDataWithHash | number | boolean | null {
  if (valueContent.slice(0, 2) === '0x') {
    return valueContent === value ? value : null;
  }

  if (!value || value === '0x') {
    return null;
  }

  return valueContentEncodingMap(valueContent).decode(value);
}
```

</file>

<file>

path: /src/lib/getData.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/lib/getData.ts

```
import { DecodeDataOutput } from '../types/decodeData';
import { GetDataDynamicKey, GetDataInput } from '../types/GetData';
import { isDynamicKeyName } from './encodeKeyName';
import {
  decodeKeyValue,
  encodeArrayKey,
  generateSchemasFromDynamicKeys,
} from './utils';
import { KeyValuePair } from '../types';
import { decodeData } from './decodeData';
import { ERC725JSONSchema } from '../types/ERC725JSONSchema';
import { ERC725Options } from '../types/Config';

/**
 * @internal
```

```
 * @param schema associated with the schema with keyType = 'Array'
 *               the data includes the raw (encoded) length key-value pair for the array
 * @param data array of key-value pairs, one of which is the length key for the schema array
 *             Data can hold other field data not relevant here, and will be ignored
 * @return an array of keys/values
 */
const getArrayValues = async (
  erc725Options: ERC725Options,
  schema: ERC725JSONSchema,
  data: Record<string, any>,
) => {
  if (schema.keyType !== 'Array') {
    throw new Error(
       The "getArrayValues" method requires a schema definition with "keyType: Array",
        ${schema} ,
    );
  }
  const results: { key: string; value }[] = [];

  // 1. get the array length
  const value = data[schema.key]; // get the length key/value pair

  if (!value || !value.value) {
    return results;
  } // Handle empty/non-existent array

  const arrayLength = await decodeKeyValue(
    'Number',
    'uint128',
    value.value,
    schema.name,
  ); // get the int array length

  const arrayElementKeys: string[] = [];
  for (let index = 0; index < arrayLength; index++) {
    const arrayElementKey = encodeArrayKey(schema.key, index);
    if (!data[arrayElementKey]) {
      arrayElementKeys.push(arrayElementKey);
    }
  }

  try {
    const arrayElements = await erc725Options.provider?.getAllData(
      erc725Options.address as string,
      arrayElementKeys,
    );

    results.push(...arrayElements);
  } catch (err) {
    // This case may happen if user requests an array key which does not exist in the contract.
    // In this case, we simply skip
  }
```

```
  return results;
};

const getDataMultiple = async (
  erc725Options: ERC725Options,
  keyNames: Array<string | GetDataDynamicKey>,
) => {
  const schemas = generateSchemasFromDynamicKeys(
    keyNames,
    erc725Options.schemas,
  );

  // Get all the raw data from the provider based on schema key hashes
  const allRawData: KeyValuePair[] = await erc725Options.provider?.getAllData(
    erc725Options.address as string,
    schemas.map((schema) => schema.key),
  );

  const keyValueMap = allRawData.reduce<{ [key: string]: any }>(
    (accumulator, current) => {
      accumulator[current.key] = current.value;
      return accumulator;
    },
    {},
  );

  const schemasWithValue = schemas.map((schema) => {
    return { ...schema, value: keyValueMap[schema.key] || null };
  });

  // ------- BEGIN ARRAY HANDLER -------
  // Get missing 'Array' fields for all arrays, as necessary

  const arraySchemas = schemas.filter(
    (e) => e.keyType.toLowerCase() === 'array',
  );

  // Looks like it gets array even if not requested as it gets the arrays from the this.options.schemas?
  // eslint-disable-next-line no-restricted-syntax
  for (const keySchema of arraySchemas) {
    const dataKeyValue = {
      [keySchema.key]: {
        key: keySchema.key,
        value: keyValueMap[keySchema.key],
      },
    };
    const arrayValues = await getArrayValues(
      erc725Options,
      keySchema,
      dataKeyValue,
    );
```

```
    if (arrayValues && arrayValues.length > 0) {
      arrayValues.push(dataKeyValue[keySchema.key]); // add the raw data array length

      schemasWithValue[
        schemasWithValue.findIndex((schema) => schema.key === keySchema.key)
      ] = { ...keySchema, value: arrayValues };
    }
  }
  // ------- END ARRAY HANDLER -------

  return decodeData(
    schemasWithValue.map(({ key, value }) => {
      return {
        keyName: key,
        value,
        // no need to add dynamic key parts here as the schemas object below already holds the "generated"
schemas for the dynamic keys
      };
    }),
    schemas,
  );
};

/**
 * Gets **decoded data** for one, many or all keys of the specified  ERC725  smart-contract.
 * When omitting the  keyOrKeys  parameter, it will get all the keys (as per {@link
ERC725JSONSchema | ERC725JSONSchema} definition).
 *
 * Data returned by this function does not contain external data of [ JSONURL ]
(https://github.com/lukso-network/LIPs/blob/master/LSPs/LSP-2-ERC725YJSONSchema.md#jsonurl)
 * or [ ASSETURL ](https://github.com/lukso-network/LIPs/blob/master/LSPs/LSP-2-
ERC725YJSONSchema.md#asseturl) schema elements.
 *
 * If you would like to receive everything in one go, you can use fetchData() from index.ts for that.
 *
 * @param {*} keyOrKeys The name (or the encoded name as the schema 'key') of the schema element in
the class instance's schema.
 *
 * @returns If the input is an array: an object with schema element key names as properties, with
corresponding **decoded** data as values. If the input is a string, it directly returns the **decoded** data.
 */

export const getData = async (
  erc725Options: ERC725Options,
  keyOrKeys?: GetDataInput,
): Promise<DecodeDataOutput | DecodeDataOutput[]> => {
  if (!keyOrKeys) {
    // eslint-disable-next-line no-param-reassign
    keyOrKeys = erc725Options.schemas
      .map((element) => element.name)
      .filter((key) => !isDynamicKeyName(key));
```

```
  }

  if (Array.isArray(keyOrKeys)) {
    return getDataMultiple(erc725Options, keyOrKeys);
  }

  const data = await getDataMultiple(erc725Options, [keyOrKeys]);

  return data[0];
};
```

</file>

<file>

path: /src/lib/getDataFromExternalSources.test.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/lib/getDataFromExternalSources.test.ts

```
/*
    This file is part of @erc725/erc725.js.
    @erc725/erc725.js is free software: you can redistribute it and/or modify
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    @erc725/erc725.js is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.
    You should have received a copy of the GNU Lesser General Public License
    along with web3.js.  If not, see <http://www.gnu.org/licenses/>.
*/

/* eslint-disable no-unused-expressions */

import { expect } from 'chai';

import { ERC725JSONSchema } from '../types/ERC725JSONSchema';

import { getDataFromExternalSources } from './getDataFromExternalSources';
import { DecodeDataOutput } from '../types/decodeData';

const IPFS_GATEWAY_MOCK = 'https://mock-ipfs.mock/ipfs/';

describe('getDataFromExternalSources', () => {
  it('should not throw if the value of a JSONURL/ASSETURL is null', async () => {
    const schemas: ERC725JSONSchema[] = [
      {
        name: 'LSP3Profile',
        key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
        keyType: 'Singleton',
```

```
      valueContent: 'JSONURL',
      valueType: 'bytes',
    },
  ];

  const dataFromChain: DecodeDataOutput[] = [
    {
      name: 'LSP3Profile',
      key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
      // @ts-ignore
      value: null,
    },
  ];

  expect(async () => {
    await getDataFromExternalSources(
      schemas,
      dataFromChain,
      IPFS_GATEWAY_MOCK,
    );
  }).to.not.throw();
 });
});
```

</file>

<file>

path: /src/lib/getDataFromExternalSources.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/lib/getDataFromExternalSources.ts

```
/*
  This file is part of @erc725/erc725.js.
  @erc725/erc725.js is free software: you can redistribute it and/or modify
  it under the terms of the GNU Lesser General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.
  @erc725/erc725.js is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU Lesser General Public License for more details.
  You should have received a copy of the GNU Lesser General Public License
  along with @erc725/erc725.js.  If not, see <http://www.gnu.org/licenses/>.
*/
/**
 * @file lib/getDataFromExternalSources.ts
 * @author Hugo Masclet <@Hugoo>
 * @author Callum Grindle <@CallumGrindle>
 * @author Reto Ryter <@rryter>
 * @date 2021
```

```
 */

import {
  DecodeDataOutput,
  GetDataExternalSourcesOutput,
} from '../types/decodeData';
import { ERC725JSONSchema } from '../types/ERC725JSONSchema';
import { SUPPORTED_VERIFICATION_METHOD_STRINGS } from '../constants/constants';
import { isDataAuthentic, patchIPFSUrlsIfApplicable } from './utils';

export const getDataFromExternalSources = (
  schemas: ERC725JSONSchema[],
  dataFromChain: DecodeDataOutput[],
  ipfsGateway: string,
): Promise<GetDataExternalSourcesOutput[]> => {
  const promises = dataFromChain.map(async (dataEntry) => {
    const schemaElement = schemas.find(
      (schema) => schema.key === dataEntry.key,
    );

    if (!schemaElement) {
      // It is weird if we can't find the schema element for the key...
      // Let's simply ignore and return it...
      return dataEntry;
    }

    if (
      !['jsonurl', 'asseturl'].includes(
        schemaElement.valueContent.toLowerCase(),
      )
    ) {
      return dataEntry;
    }

    // At this stage, value should be of type jsonurl or asseturl
    if (typeof dataEntry.value === 'string') {
      console.error(
         Value of key: ${dataEntry.name} (${dataEntry.value}) is string but valueContent is:
${schemaElement.valueContent}. Expected type should be object with url key. ,
      );
      return dataEntry;
    }

    if (!dataEntry.value) {
      return dataEntry;
    }

    if (Array.isArray(dataEntry.value)) {
      console.error(
         Value of key: ${dataEntry.name} (${dataEntry.value}) is string[] but valueContent is:
${schemaElement.valueContent}. Expected type should be object with url key. ,
      );
```

```
    return dataEntry;
  }

  const urlDataWithHash = dataEntry.value; // Type URLDataWithHash

  let receivedData;
  try {
    const { url } = patchIPFSUrlsIfApplicable(urlDataWithHash, ipfsGateway);

    receivedData = await fetch(url).then(async (response) => {
      if (
        urlDataWithHash.verification?.method ===
        SUPPORTED_VERIFICATION_METHOD_STRINGS.KECCAK256_BYTES
      ) {
        return response
          .arrayBuffer()
          .then((buffer) => new Uint8Array(buffer));
      }

      return response.json();
    });
  } catch (error) {
    console.error(error,  GET request to ${urlDataWithHash.url} failed );
    throw error;
  }

  return isDataAuthentic(receivedData, urlDataWithHash.verification)
    ? { ...dataEntry, value: receivedData }
    : { ...dataEntry, value: null };
 });

 return Promise.all(promises);
};
```

</file>

<file>

path: /src/lib/getSchemaElement.test.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/lib/getSchemaElement.test.ts

```
/*
  This file is part of @erc725/erc725.js.
  @erc725/erc725.js is free software: you can redistribute it and/or modify
  it under the terms of the GNU Lesser General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.
  @erc725/erc725.js is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
```

```
   GNU Lesser General Public License for more details.
   You should have received a copy of the GNU Lesser General Public License
   along with web3.js.  If not, see <http://www.gnu.org/licenses/>.
*/

import assert from 'assert';

import { ERC725JSONSchema } from '../types/ERC725JSONSchema';
import { getSchemaElement } from './getSchemaElement';

describe('getSchemaElement', () => {
  it('gets the schemaElement from key name and key hash (with and without 0x prefix) correctly', () => {
    const schemas: ERC725JSONSchema[] = [
      {
        name: 'LSP3Profile',
        key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
        keyType: 'Singleton',
        valueContent: 'JSONURL',
        valueType: 'bytes',
      },
    ];

    assert.strictEqual(getSchemaElement(schemas, schemas[0].name), schemas[0]);
    assert.strictEqual(getSchemaElement(schemas, schemas[0].key), schemas[0]);
    assert.strictEqual(
      getSchemaElement(schemas, schemas[0].key.slice(2)),
      schemas[0],
    );
  });

  const schemasWithDynamicKey: ERC725JSONSchema[] = [
    {
      name: 'LSP12IssuedAssetsMap:<address>',
      key: '0x74ac2555c10b9349e78f0000<address>',
      keyType: 'Mapping',
      valueType: 'bytes',
      valueContent: 'Mixed',
    },
    {
      name: 'ARandomKey',
      key: '0x7cf0c8053453d0353fdbad6a48e68966b35dd13cb3a62e7b75009dc5035b80c0',
      keyType: 'Singleton',
      valueContent: 'JSONURL',
      valueType: 'bytes',
    },
  ];

  it('throws is attempt to get a dynamic key without dynamicKeyParts', () => {
    assert.throws(() =>
      getSchemaElement(schemasWithDynamicKey, 'LSP12IssuedAssetsMap:<address>'),
    );
  });
```

```
  it('gets the schemeElement for a dynamic key correctly', () => {
    assert.deepStrictEqual(
      getSchemaElement(
        schemasWithDynamicKey,
        'LSP12IssuedAssetsMap:<address>',
        ['0x2ab3903c6e5815f4bc2a95b7f3b22b6a289bacac'],
      ),
      {
        name: 'LSP12IssuedAssetsMap:2ab3903c6e5815f4bc2a95b7f3b22b6a289bacac',
        key: '0x74ac2555c10b9349e78f00002ab3903c6e5815f4bc2a95b7f3b22b6a289bacac',
        keyType: 'Mapping',
        valueType: 'bytes',
        valueContent: 'Mixed',
      },
    );
  });
});
```

</file>

<file>

path: /src/lib/getSchemaElement.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/lib/getSchemaElement.ts

```
/*
    This file is part of @erc725/erc725.js.
    @erc725/erc725.js is free software: you can redistribute it and/or modify
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    @erc725/erc725.js is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.
    You should have received a copy of the GNU Lesser General Public License
    along with @erc725/erc725.js.  If not, see <http://www.gnu.org/licenses/>.
*/
/**
 * @file lib/getSchemaElement.ts
 * @author Hugo Masclet <@Hugoo>
 * @date 2021
 */

import { isHex, isHexStrict } from 'web3-utils';
import { DynamicKeyParts } from '../types/dynamicKeys';
import { ERC725JSONSchema } from '../types/ERC725JSONSchema';
import {
  encodeKeyName,
```

```
  generateDynamicKeyName,
  isDynamicKeyName,
} from './encodeKeyName';

/**
 *
 * @param schemas
 * @param namedDynamicKey
 * @param dynamicKeyParts
 * @returns
 */
const getSchemaElementForDynamicKeyName = (
  schemas: ERC725JSONSchema[],
  namedDynamicKey: string,
  dynamicKeyParts: DynamicKeyParts,
): ERC725JSONSchema => {
  // In that case, we will generate a new schema element with the final computed name and encoded key
hash.

  const schemaElement = schemas.find((e) => e.name === namedDynamicKey);

  if (!schemaElement) {
    throw new Error(
       No matching schema found for dynamic key: ${namedDynamicKey} ,
    );
  }

  // once we have the schemaElement with dynamic parts, we need to replace the name and the key:

  const key = encodeKeyName(namedDynamicKey, dynamicKeyParts);
  const name = generateDynamicKeyName(namedDynamicKey, dynamicKeyParts);

  return {
    ...schemaElement,
    key,
    name,
  };
};

/**
 *
 * @param schemas An array of ERC725JSONSchema objects.
 * @param {string} namedOrHashedKey A string of either the schema element name, or hashed key (with or
without the 0x prefix).
 * @param dynamicKeyParts if a dynamic named key is given, you should also set the dynamicKeyParts.
 *
 * @return The requested schema element from the full array of schemas.
 */
export function getSchemaElement(
  schemas: ERC725JSONSchema[],
  namedOrHashedKey: string,
  dynamicKeyParts?: DynamicKeyParts,
```

```
): ERC725JSONSchema {
  let keyHash: string;

  if (isDynamicKeyName(namedOrHashedKey)) {
    if (!dynamicKeyParts) {
      throw new Error(
         Can't getSchemaElement for dynamic key: ${namedOrHashedKey} without
dynamicKeyParts. ,
      );
    }
    return getSchemaElementForDynamicKeyName(
      schemas,
      namedOrHashedKey,
      dynamicKeyParts,
    );
  }

  if (isHex(namedOrHashedKey)) {
    keyHash = isHexStrict(namedOrHashedKey)
      ? namedOrHashedKey
      :  0x${namedOrHashedKey} ;
  } else {
    keyHash = encodeKeyName(namedOrHashedKey);
  }

  const schemaElement = schemas.find((e) => e.key === keyHash);

  if (!schemaElement) {
    throw new Error(
       No matching schema found for key: ${namedOrHashedKey} (${keyHash}). ,
    );
  }

  return schemaElement;
}
```

</file>

<file>

path: /src/lib/isValidSignature.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/lib/isValidSignature.ts

```
/*
  This file is part of @erc725/erc725.js.
  @erc725/erc725.js is free software: you can redistribute it and/or modify
  it under the terms of the GNU Lesser General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.
  @erc725/erc725.js is distributed in the hope that it will be useful,
```

```
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU Lesser General Public License for more details.
   You should have received a copy of the GNU Lesser General Public License
   along with @erc725/erc725.js.  If not, see <http://www.gnu.org/licenses/>.
*/
/**
 * @file lib/isValidSignature.ts
 * @author Hugo Masclet <@Hugoo>
 * @date 2022
 */

import { keccak256 } from 'web3-utils';
import { ProviderWrapper } from '../provider/providerWrapper';

const MAGIC_VALUE = '0x1626ba7e';

// https://ethereum.stackexchange.com/a/72625
function validateHash(hash) {
  return /^0x([A-Fa-f0-9]{64})$/.test(hash);
}

/**
 *
 * https://eips.ethereum.org/EIPS/eip-1271
 *
 * @param {string} messageOrHash
 * @param {string} signature
 * @param {string} address the contract address
 * @returns {Promise<boolean>} Return true if the signature is valid (if the contract returns the magic value),
false if not.
 */
export const isValidSignature = async (
  messageOrHash: string,
  signature: string,
  address: string,
  wrappedProvider: ProviderWrapper,
): Promise<boolean> => {
  const hash = validateHash(messageOrHash)
    ? messageOrHash
    : keccak256(
         \x19Ethereum Signed Message:\n${messageOrHash.length}${messageOrHash} ,
      );

  if (signature.length !== 132) {
    throw new Error('Signature length should be 132 (65bytes)');
  }

  try {
    const value = await wrappedProvider.isValidSignature(
      address,
      hash,
```

```
      signature,
    );

    return value === MAGIC_VALUE;
  } catch (err: any) {
    throw new Error(
      &amp;nbsp;Error when checking signature. Is ${address} a valid contract address which supports EIP-1271 standard?&amp;nbsp;,
    );
  }
};
```

</file>

<file>

path: /src/lib/provider-wrapper-utils.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/lib/provider-wrapper-utils.ts

```
/*
    This file is part of @erc725/erc725.js.
    @erc725/erc725.js is free software: you can redistribute it and/or modify
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    @erc725/erc725.js is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.
    You should have received a copy of the GNU Lesser General Public License
    along with @erc725/erc725.js.  If not, see <http://www.gnu.org/licenses/>.
*/

import * as abi from 'web3-eth-abi';
import { numberToHex } from 'web3-utils';

import {
  JsonRpc,
  JsonRpcEthereumProviderParamsWithLatest,
} from '../types/JsonRpc';
import { Method } from '../types/Method';

import { METHODS } from '../constants/constants';

let idCount = 0;
const web3abiDecoder = abi.default;

export function decodeResult(method: Method, hexString: string) {
  if (hexString === '0x') {
    return null;
```

```
  }

  const decodedData = web3abiDecoder.decodeParameter(
    METHODS[method].returnEncoding,
    hexString,
  );

  if (
    Array.isArray(decodedData) &&
    decodedData.length === 1 &&
    decodedData[0] === '0x'
  ) {
    return [null];
  }

  return decodedData;
}

const constructJSONRPCParams = (
  address: string,
  method: Method,
  gasInfo: number,
  methodParam?: string,
): JsonRpcEthereumProviderParamsWithLatest => {
  const data = methodParam
    ? METHODS[method].sig + methodParam.replace('0x', '')
    : METHODS[method].sig;

  return [
    {
      to: address,
      value: METHODS[method].value,
      gas: numberToHex(gasInfo),
      data,
    },
    'latest',
  ];
};

export function constructJSONRPC(
  address: string,
  method: Method,
  gasInfo: number,
  methodParam?: string,
): JsonRpc {
  idCount += 1;
  return {
    jsonrpc: '2.0',
    method: 'eth_call',
    params: constructJSONRPCParams(address, method, gasInfo, methodParam),
    id: idCount,
  };
```

```
}
```

</file>

<file>

path: /src/lib/schemaParser.test.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/lib/schemaParser.test.ts

```
/*
    This file is part of @erc725/erc725.js.
    @erc725/erc725.js is free software: you can redistribute it and/or modify
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    @erc725/erc725.js is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.
    You should have received a copy of the GNU Lesser General Public License
    along with web3.js.  If not, see <http://www.gnu.org/licenses/>.
*/

import assert from 'assert';
import { ERC725JSONSchema } from '../types/ERC725JSONSchema';

import { getSchema } from './schemaParser';

describe('schemaParser getSchema', () => {
  describe('Singleton', () => {
    it('finds keys of type Singleton correctly', () => {
      const schema = getSchema(
        '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
      );

      assert.deepStrictEqual(schema, {
        name: 'LSP3Profile',
        key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
        keyType: 'Singleton',
        valueContent: 'JSONURL',
        valueType: 'bytes',
      });
    });
  });

  describe('Array', () => {
    it('finds initial key of type Array correctly', () => {
      const schema = getSchema(
        '0x7c8c3416d6cda87cd42c71ea1843df28ac48503544f988d55ee2eaa47b6dc05cd',
      );
```

```
    assert.deepStrictEqual(schema, {
      name: 'LSP12IssuedAssets[]',
      key: '0x7c8c3416d6cda87cd42c71ea1843df28ac4850354f988d55ee2eaa47b6dc05cd',
      keyType: 'Array',
      valueContent: 'Address',
      valueType: 'address',
    });
  });
  it('finds subsequent key of type Array correctly', () => {
    const schema = getSchema([
      '0x7c8c3416d6cda87cd42c71ea1843df2800000000000000000000000000000001',
      '0xdf30dba06db6a30e65354d9a64c609860000000000000000000000000000000',
    ]);

    assert.deepStrictEqual(schema, {
      '0x7c8c3416d6cda87cd42c71ea1843df2800000000000000000000000000000001': {
        name: 'LSP12IssuedAssets[1]',
        key: '0x7c8c3416d6cda87cd42c71ea1843df2800000000000000000000000000000001',
        keyType: 'Singleton',
        valueContent: 'Address',
        valueType: 'address',
      },
      '0xdf30dba06db6a30e65354d9a64c6098600000000000000000000000000000000': {
        name: 'AddressPermissions[0]',
        key: '0xdf30dba06db6a30e65354d9a64c6098600000000000000000000000000000000',
        keyType: 'Singleton',
        valueContent: 'Address',
        valueType: 'address',
      },
    });
  });
  it('finds subsequent key of type Array correctly', () => {
    const schema = getSchema(
      '0x3a47ab5bd3a594c3a8995f8fa58d087600000000000fab0000000000000000001',
    );

    assert.deepStrictEqual(schema, null);
  });
});

describe('Mapping', () => {
  it('finds known mappings', () => {
    const schema = getSchema(
      '0xeafec4d89fa9619884b600005ef83ad9559033e6e941db7d7c495acdce616347',
    );

    assert.deepStrictEqual(schema, {
      name: 'SupportedStandards:LSP3Profile',
      key: '0xeafec4d89fa9619884b600005ef83ad9559033e6e941db7d7c495acdce616347',
      keyType: 'Mapping',
      valueContent: '0x5ef83ad9',
```

```
      valueType: 'bytes4',
    });
  });

  it('finds unknown mappings', () => {
    // Key name: SupportedStandards:UnknownKey
    const schema = getSchema(
      '0xeafec4d89fa9619884b60000f4d7faed14a1ab658d46d385bc29fb1eeaa56d0b',
    );

    assert.deepStrictEqual(schema, {
      name: 'SupportedStandards:??????',
      key: '0xeafec4d89fa9619884b60000f4d7faed14a1ab658d46d385bc29fb1eeaa56d0b',
      keyType: 'Mapping',
      valueContent: '?',
      valueType: 'bytes4',
    });
  });
  it('finds Known Mapping:<address> ', () => {
    const address = 'af3bf2ffb025098b79caddfbdd113b3681817744';
    const name = &amp;nbsp;MyCoolAddress:${address}&amp;nbsp;;
    const key = &amp;nbsp;0x22496f48a493035f00000000${address}&amp;nbsp;;

    const extraSchema: ERC725JSONSchema = {
      name,
      key,
      keyType: 'Mapping',
      valueContent: 'Address',
      valueType: 'address',
    };

    const schema = getSchema(key, [extraSchema]);

    assert.deepStrictEqual(schema, extraSchema);
  });
});

describe('MappingWithGrouping', () => {
  it('finds MappingWithGrouping', () => {
    const address = 'af3bf2ffb025098b79caddfbdd113b3681817744';
    const name = &amp;nbsp;AddressPermissions:Permissions:${address}&amp;nbsp;;
    const key = &amp;nbsp;0x4b80742de2bf82acb3630000${address}&amp;nbsp;;
    const schema = getSchema(key);

    assert.deepStrictEqual(schema, {
      name,
      key,
      keyType: 'MappingWithGrouping',
      valueContent: 'BitArray',
      valueType: 'bytes32',
    });
  });
```

```
  });
});
```

</file>

<file>

path: /src/lib/schemaParser.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/lib/schemaParser.ts

```
/*
    This file is part of @erc725/erc725.js.
    @erc725/erc725.js is free software: you can redistribute it and/or modify
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    @erc725/erc725.js is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.
    You should have received a copy of the GNU Lesser General Public License
    along with @erc725/erc725.js.  If not, see <http://www.gnu.org/licenses/>.
*/
/**
 * @file lib/schemaParser.ts
 * @author Hugo Masclet <@Hugoo>
 * @date 2022
 */

import allSchemas from '../schemas';

import {
  ERC725JSONSchema,
  ERC725JSONSchemaKeyType,
} from '../types/ERC725JSONSchema';

const getSchemasByKeyType = (
  schemas: ERC725JSONSchema[],
): Record<ERC725JSONSchemaKeyType, ERC725JSONSchema[]> => {
  return {
    Singleton: schemas.filter((schema) => schema.keyType === 'Singleton'),
    Array: schemas.filter((schema) => schema.keyType === 'Array'),
    Mapping: schemas.filter((schema) => schema.keyType === 'Mapping'),
    MappingWithGrouping: schemas.filter(
      (schema) => schema.keyType === 'MappingWithGrouping',
    ),
  };
};

const findSingletonSchemaForKey = (
```

```
  key: string,
  schemas: ERC725JSONSchema[],
): ERC725JSONSchema | null => {
  return schemas.find((schema) => schema.key === key) || null;
};

const findArraySchemaForKey = (
  key: string,
  schemas: ERC725JSONSchema[],
): ERC725JSONSchema | null => {
  // Should detect:

  // 1. Initial key
  const initialKeySchema = schemas.find((schema) => schema.key === key) || null;

  if (initialKeySchema) {
    return initialKeySchema;
  }

  // 2. Subsequent keys
  const bytes16Key = key.substring(0, 34);
  const arraySchema =
    schemas.find((schema) => schema.key.substring(0, 34) === bytes16Key) ||
    null;

  if (!arraySchema) {
    return null;
  }

  // https://stackoverflow.com/a/1779019/651299
  if (!/^\d+$/.test(key.substring(34))) {
    return null;
  }

  const elementIndex = parseInt(key.substring(34), 10);

  return {
    ...arraySchema,
    key,
    name: arraySchema.name.replace('[]',  [${elementIndex}] ),
    keyType: 'Singleton',
  };
};

const findMappingSchemaForKey = (
  key: string,
  schemas: ERC725JSONSchema[],
): ERC725JSONSchema | null => {
  // Should detect:

  // 1. Known/defined mapping
  let keySchema = schemas.find((schema) => schema.key === key) || null;
```

```
  if (keySchema) {
    return keySchema;
  }

  // 2. "Semi defined mappings" i.e. "SupportedStandards:??????"
  keySchema =
    schemas.find(
      (schema) =>  ${schema.key.substring(0, 22)}0000  === key.substring(0, 26),
    ) || null;

  if (!keySchema) {
    return null;
  }
  // TODO: Handle the SupportedStandard Keys; we can get the valueContent from the Keys
  return {
    ...keySchema,
    valueContent: '?',
    name:  ${keySchema.name.split(':')[0]}:?????? ,
    key,
  };
};

const findMappingWithGroupingSchemaForKey = (
  key: string,
  schemas: ERC725JSONSchema[],
): ERC725JSONSchema | null => {
  const keySchema =
    schemas.find(
      (schema) => schema.key.substring(0, 26) === key.substring(0, 26),
    ) || null;

  const address = key.substring(26);

  if (keySchema) {
    return {
      ...keySchema,
      key,
      name:  ${keySchema.name.substring(
        0,
        keySchema.name.lastIndexOf(':'),
      )}:${address} ,
    };
  }

  return null;
};

function schemaParser(
  key: string,
  schemas: ERC725JSONSchema[],
): ERC725JSONSchema | null {
```

```
  const schemasByKeyType = getSchemasByKeyType(schemas);

  let foundSchema: ERC725JSONSchema | null = null;

  foundSchema = findSingletonSchemaForKey(key, schemasByKeyType.Singleton);

  if (foundSchema) {
    return foundSchema;
  }

  foundSchema = findArraySchemaForKey(key, schemasByKeyType.Array);

  if (foundSchema) {
    return foundSchema;
  }

  foundSchema = findMappingSchemaForKey(key, schemasByKeyType.Mapping);

  if (foundSchema) {
    return foundSchema;
  }

  foundSchema = findMappingWithGroupingSchemaForKey(
    key,
    schemasByKeyType.MappingWithGrouping,
  );

  return foundSchema;
}

export function getSchema(
  keyOrKeys: string | string[],
  providedSchemas?: ERC725JSONSchema[],
): ERC725JSONSchema | null | Record<string, ERC725JSONSchema | null> {
  let fullSchema: ERC725JSONSchema[] = allSchemas;
  if (providedSchemas) {
    fullSchema = fullSchema.concat(providedSchemas);
  }

  if (Array.isArray(keyOrKeys)) {
    return keyOrKeys.reduce<Record<string, ERC725JSONSchema | null>>(
      (acc, key) => {
        acc[key] = schemaParser(key, fullSchema);
        return acc;
      },
      {},
    );
  }

  return schemaParser(keyOrKeys, fullSchema);
}
```

</file>

<file>

path: /src/lib/utils.test.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/lib/utils.test.ts

```
/*
    This file is part of @erc725/erc725.js.
    @erc725/erc725.js is free software: you can redistribute it and/or modify
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    @erc725/erc725.js is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.
    You should have received a copy of the GNU Lesser General Public License
    along with web3.js.  If not, see <http://www.gnu.org/licenses/>.
*/

/* eslint-disable no-unused-expressions */

import { expect } from 'chai';
import assert from 'assert';

import { keccak256, utf8ToHex } from 'web3-utils';
import {
  ERC725JSONSchema,
  ERC725JSONSchemaKeyType,
  ERC725JSONSchemaValueType,
} from '../types/ERC725JSONSchema';
import { GetDataDynamicKey } from '../types/GetData';

import { SUPPORTED_VERIFICATION_METHOD_STRINGS } from '../constants/constants';
import {
  guessKeyTypeFromKeyName,
  isDataAuthentic,
  encodeArrayKey,
  encodeKeyValue,
  decodeKeyValue,
  encodeKey,
  encodeData,
  convertIPFSGatewayUrl,
  generateSchemasFromDynamicKeys,
  encodeTupleKeyValue,
} from './utils';
import { isDynamicKeyName } from './encodeKeyName';
import { decodeKey } from './decodeData';
```

```
describe('utils', () => {
  describe('encodeKey/decodeKey', () => {
    const testCases = [
      {
        schema: {
          name: 'LSP3IssuedAssets[]',
          key: '0x3a47ab5bd3a594c3a8995f8fa58d0876c96819ca4516bd76100c92462f2f9dc0',
          keyType: 'Array',
          valueContent: 'Address',
          valueType: 'address',
        },
        decodedValue: [
          '0xc444009d38d3046bb0cF81Fa2Cd295ce46A67C78',
          '0x4fEbC3491230571F6e1829E46602e3b110215A2E',
        ],
        encodedValue: [
          {
            key: '0x3a47ab5bd3a594c3a8995f8fa58d0876c96819ca4516bd76100c92462f2f9dc0',
            value: '0x00000000000000000000000000000002',
          },
          {
            key: '0x3a47ab5bd3a594c3a8995f8fa58d087600000000000000000000000000000000',
            value: '0xc444009d38d3046bb0cf81fa2cd295ce46a67c78',
          },
          {
            key: '0x3a47ab5bd3a594c3a8995f8fa58d087600000000000000000000000000000001',
            value: '0x4febc3491230571f6e1829e46602e3b110215a2e',
          },
        ],
      },
      {
        schema: {
          name: 'TestObjArray[]',
          key: '0x9985edaf12cbacf5ac7d6ed54f0445cc0ea56075aee9b9942e4ab3bf4239f950',
          keyType: 'Array',
          valueContent: 'JSONURL',
          valueType: 'bytes',
        },
        decodedValue: [
          {
            verification: {
              method: SUPPORTED_VERIFICATION_METHOD_STRINGS.KECCAK256_UTF8,
              data: '0x733e78f2fc4a3304c141e8424d02c9069fe08950c6514b27289ead8ef4faa49d',
            },
            url: 'ipfs://QmbErKh3FjsAR6YjsTjHZNm6McDp6aRt82Ftcv9AJJvZbd',
          },
          {
            verification: {
              method: SUPPORTED_VERIFICATION_METHOD_STRINGS.KECCAK256_UTF8,
              data: '0x81bd0b7ed5ac354abbf24619ce169333f00a4bdfa8fcaf3791d25f69b497abf88',
            },
            url: 'ipfs://QmbErKh3Fjsxxxxxxxxxxxxxxxxxxxxxxxxxxxxv9AJJvZbd',
```

```
        },
      ],
      encodedValue: [
        {
          key: '0x9985edaf12cbacf5ac7d6ed54f0445cc0ea56075aee9b9942e4ab3bf4239f950',
          value: '0x00000000000000000000000000000002',
        },
        {
          key: '0x9985edaf12cbacf5ac7d6ed54f0445cc00000000000000000000000000000000',
          value:

'0x6f357c6a733e78f2fc4a3304c141e8424d02c9069fe08950c6514b27289ead8ef4faa49d6970667
33a2f2f516d6245724b6833466a73415236596a73546a485a4e6d364d6344703661527438324674637
639414a4a765a6264',
        },
        {
          key: '0x9985edaf12cbacf5ac7d6ed54f0445cc00000000000000000000000000000001',
          value:

'0x6f357c6a81bd0b7ed5ac354abbf24619ce16933f00a4bdfa8fcaf3791d25f69b497abf8869706673
3a2f2f516c6245724b6833466a737878787878787878787878787878787878787878787878787878
7639414a4a765a6264',
        },
      ],
    },
    {
      schema: {
        name: 'TupleType',
        key: '0x50f2bd37dff422565ef30f885bf0ee0a36a2a2bcf9b0146ac9c646679c38dfd8',
        keyType: 'Singleton',
        valueType: '(bytes4,bytes8)',
        valueContent: '(Bytes4,Number)',
      },
      decodedValue: ['0xcafecafe', 11],
      encodedValue: '0xcafecafe000000000000000b',
    },
    {
      schema: {
        name: 'TupleMultiType',
        key: '1e1bc4abe01b7baa7d4a359c0f460e632ef34b3f16f5722bd8892f2dae913022',
        keyType: 'Singleton',
        valueType: '(bytes4,bytes8,bytes4)',
        valueContent: '(Bytes4,Number,Number)',
      },
      decodedValue: ['0xcafecafe', 11, 8],
      encodedValue: '0xcafecafe000000000000000b00000008',
    },
    {
      schema: {
        name: 'TupleMultiType',
        key: '1e1bc4abe01b7baa7d4a359c0f460e632ef34b3f16f5722bd8892f2dae913022',
        keyType: 'Singleton',
```

```
      valueType: '(bytes4,address,bytes2)[CompactBytesArray]',
      valueContent: '(Bytes4,Address,Bytes)',
    },
    decodedValue: [
      [
        '0xcafecafe',
        '0xDAFEA492D9c6733ae3d56b7Ed1ADB60692c98Bc5',
        '0xcafe',
      ],
      [
        '0xbeefbeef',
        '0xFE313200faF8Da1492Eadf8Deb79bd264D7cF2141',
        '0xbeef',
      ],
      [
        '0xf00df00d',
        '0xc527702b14BF2f79F70B32e09F62B6A74cADFd80',
        '0xf00d',
      ],
    ],
    encodedValue:

'0x001acafecafedafea492d9c6733ae3d56b7ed1adb60692c98bc5cafe001abeefbeeffe313200faf8da1492eadf
8deb79bd264d7cf2141beef001af00df00dc527702b14bf2f79f70b32e09f62b6a74cadfd80f00d',
  },
  {
    schema: {
      name: 'TupleMultiType',
      key: '1e1bc4abe01b7baa7d4a359c0f460e632ef34b3f16f5722bd8892f2dae913022',
      keyType: 'Singleton',
      valueType: '(bytes4,bytes8,bytes16)[CompactBytesArray]',
      valueContent: '(Bytes4,Bytes8,Bytes16)',
    },
    decodedValue: [
      [
        '0xcafecafe',
        '0x951a5d121531bba8',
        '0xdafea492d9c6733ae3d56b7ed1adb606',
      ],
      [
        '0xbeefbeef',
        '0x8a483080f5db1105',
        '0xfe313200faf8da1492eadf8deb79bd264',
      ],
      [
        '0xf00df00d',
        '0x2fe92a11caf28ab2',
        '0xc527702b14bf2f79f70b32e09f62b6a7',
      ],
    ],
    encodedValue:
```

```
'0x001ccafecafe951a5d121531bba8dafea492d9c6733ae3d56b7ed1adb606001cbeefbeef8a483080f5db1105
fe31320faf8da1492eadf8deb79bd264001cf00df00d2fe92a11caf28ab2c527702b14bf2f79f70b32e09f62b6a7'
,
      },
      {
        schema: {
          name: 'AddressPermissions:AllowedCalls:<address>',
          key: '0x4b80742de2bf393a64c70000<address>',
          keyType: 'MappingWithGrouping',
          valueType: '(bytes4,address,bytes4)[CompactBytesArray]',
          valueContent: '(Bytes4,Address,Bytes4)',
        },
        decodedValue: [
          [
            '0xcafecafe',
            '0xDAFEA492D9c6733ae3d56b7Ed1ADB60692c98Bc5',
            '0xcafecafe',
          ],
          [
            '0xbeefbeef',
            '0xFE31320faF8Da1492Eadf8Deb79bd264D7cF2141',
            '0xbeefbeef',
          ],
          [
            '0xf00df00d',
            '0xc527702b14BF2f79F70B32e09F62B6A74cADFd80',
            '0xf00df00d',
          ],
        ],
        encodedValue:

'0x001ccafecafeDAFEA492D9c6733ae3d56b7Ed1ADB60692c98Bc5cafecafe001cbeefbeefFE31320faF8Da
1492Eadf8Deb79bd264D7cF2141beefbeef001cf00df00dc527702b14BF2f79F70B32e09F62B6A74cADFd80
f00df00d'.toLowerCase(),
      },
    ];

    testCases.forEach((testCase) => {
      it(&amp;nbsp;encodes/decodes keyType Array / tuples (valueContent: ${testCase.schema.valueContent},
valueType: ${testCase.schema.valueType}&amp;nbsp;, () => {
        assert.deepStrictEqual(
          encodeKey(testCase.schema as ERC725JSONSchema, testCase.decodedValue),
          testCase.encodedValue,
        );
        assert.deepStrictEqual(
          decodeKey(testCase.schema as ERC725JSONSchema, testCase.encodedValue),
          testCase.decodedValue,
        );
      });
    });
  });
```

```
describe('encodeKeyValue/decodeKeyValue', () => {
  const testCases = [
    {
      valueContent: 'Keccak256',
      valueType: 'bytes32[]',
      decodedValue: [
        '0xe5d35cae7c9db9879eb8a205baa046ad99503414d6a55eb6725494a4254a6d3f',
        '0x828e919feac2ec05939abd5d221692fbe6bac5667ba5af5d191df1f7ecb1ac21',
      ],
      encodedValue:

'0x00000000000000000000000000000000000000000000000000000000000000200000000000000000000000
0000000000000000000000000000000000000000002e5d35cae7c9db9879eb8a205baa046ad99503414d6
a55eb6725494a4254a6d3f828e919feac2ec05939abd5d221692fbe6bac5667ba5af5d191df1f7ecb1ac21',
    },
    {
      valueContent: 'Number',
      valueType: 'uint256[]',
      decodedValue: [123, 456],
      encodedValue:

'0x00000000000000000000000000000000000000000000000000000000000000200000000000000000000000
00000000000000000000000000000000000000000020000000000000000000000000000000000000000000
000000000000000000007b00000000000000000000000000000000000000000000000000000000000001c8'
,
    },
    {
      valueContent: 'Address',
      valueType: 'address[]',
      decodedValue: [
        '0xCE3e75A43B0A29292219926EAdC8C5585651219C',
        '0xba61a0b24a228807f23B46064773D28Fe51dA81C',
      ],
      encodedValue:

'0x00000000000000000000000000000000000000000000000000000000000000200000000000000000000000
0000000000000000000000000000000000000000002000000000000000000000000ce3e75a43b0a292922
19926eadc8c5585651219c000000000000000000000000ba61a0b24a228807f23b46064773d28fe51da81c',
    },
    {
      valueContent: 'String',
      valueType: 'string[]',
      decodedValue: ['apple sauce', 'butter chicken'],
      encodedValue:

'0x00000000000000000000000000000000000000000000000000000000000000200000000000000000000000
00000000000000000000000000000000000000000020000000000000000000000000000000000000000000
000000000000000000004000000000000000000000000000000000000000000000000000000000000080
0000000000000000000000000000000000000000000000000000000000000000b6170706c65207361756365
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000e62757474657220636869636b656e000000000000000000000000000000000000',
    },
```

```
    {
      valueContent: 'String',
      valueType: 'string',
      decodedValue: 'Great-string',
      encodedValue: utf8ToHex('Great-string'),
    },
    {
      valueContent: 'Markdown',
      valueType: 'string',
      decodedValue: '# Title',
      encodedValue: utf8ToHex('# Title'),
    },
    {
      valueContent: 'URL',
      valueType: 'bytes',
      decodedValue: 'http://day.night',
      encodedValue: '0x687474703a2f2f6461792e6e69676874',
    },
    {
      valueContent: 'AssetURL',
      valueType: 'bytes',
      decodedValue: {
        verification: {
          method: SUPPORTED_VERIFICATION_METHOD_STRINGS.KECCAK256_UTF8,
          data: '0x81dadadadadadadadadadadadadadf00a4bdfa8fcaf3791d25f69b497abf88',
        },
        url: 'http://day.night/asset.glb',
      },
      encodedValue:

'0x6f357c6a81dadadadadadadadadadadadadadf00a4bdfa8fcaf3791d25f69b497abf88687474703a2f2f6461792e6e696768742f61737365742e676c62',
    },
    {
      valueContent: 'JSONURL',
      valueType: 'bytes',
      decodedValue: {
        verification: {
          method: SUPPORTED_VERIFICATION_METHOD_STRINGS.KECCAK256_UTF8,
          data: '0x81bd0b7ed5ac354abbf24619ce16933f00a4bdfa8fcaf3791d25f69b497abf88',
        },
        url: 'ipfs://QmbErKh3Fjsxxxxxxxxxxxxxxxxxxxxxxxxxxxv9AJJvZbd',
      },
      encodedValue:

'0x6f357c6a81bd0b7ed5ac354abbf24619ce16933f00a4bdfa8fcaf3791d25f69b497abf886970667333a2f2f516d6245724b6833466a737878787878787878787878787878787878787878787878787878787639414a4a765a6264',
    },

    {
      valueContent: 'Bytes',
```

```
      valueType: 'bytes',
      decodedValue: '0xaaAE32',
      encodedValue: '0xaaAE32',
    },
    {
      valueContent: 'Bytes32',
      valueType: 'bytes',
      decodedValue:
        '0x7465737400000000000000000000000000000000000000000000000000000000',
      encodedValue:
        '0x7465737400000000000000000000000000000000000000000000000000000000',
    },
    {
      valueContent: 'Bytes4',
      valueType: 'bytes',
      decodedValue: '0x74657374',
      encodedValue: '0x74657374',
    },
    {
      valueContent: '0xc9aaAE3201F40fd0fF04D9c885769d8256A456ab',
      valueType: 'bytes',
      decodedValue: '0xc9aaAE3201F40fd0fF04D9c885769d8256A456ab',
      encodedValue: '0xc9aaAE3201F40fd0fF04D9c885769d8256A456ab',
    },
  ];

  testCases.forEach((testCase) => {
    it(&amp;nbsp;encodes correctly valueContent ${testCase.valueContent} to valueType:
${testCase.valueType}&amp;nbsp;, () => {
      assert.strictEqual(
        encodeKeyValue(
          testCase.valueContent,
          testCase.valueType as ERC725JSONSchemaValueType,
          testCase.decodedValue,
        ),
        testCase.encodedValue,
      );
    });
    it(&amp;nbsp;decodes correctly valueContent: ${testCase.valueContent} to valueType:
${testCase.valueType}&amp;nbsp;, () => {
      assert.deepStrictEqual(
        decodeKeyValue(
          testCase.valueContent,
          testCase.valueType as ERC725JSONSchemaValueType,
          testCase.encodedValue,
        ),
        testCase.decodedValue,
      );
    });
  });
});
```

```
describe('encodeTupleKeyValue', () => {
  const testCases = [
    {
      valueContent: '(Bytes4,Number)',
      valueType: '(bytes4,bytes8)',
      encodedValue: '0xdeadbeaf0000000000000010',
      decodedValue: ['0xdeadbeaf', 16],
    },
  ]; // we may need to add more test cases! Address, etc.

  testCases.forEach((testCase) => {
    it(&amp;nbsp;encodes tuple values&amp;nbsp;, () => {
      expect(
        encodeTupleKeyValue(
          testCase.valueContent,
          testCase.valueType,
          testCase.decodedValue,
        ),
      ).to.eq(testCase.encodedValue);
    });
  });
});

describe('encodeArrayKey', () => {
  it('encodes array key correctly', () => {
    const key =
      '0x3a47ab5bd3a594c3a8995f8fa58d0876c96819ca4516bd76100c92462f2f9dc0';

    const expectedValues = [
      '0x3a47ab5bd3a594c3a8995f8fa58d0876000000000000000000000000000000000',
      '0x3a47ab5bd3a594c3a8995f8fa58d0876000000000000000000000000000000001',
      '0x3a47ab5bd3a594c3a8995f8fa58d0876000000000000000000000000000000002',
    ];

    expectedValues.forEach((expectedValue, index) => {
      assert.strictEqual(encodeArrayKey(key, index), expectedValue);
    });
  });
});

describe('encodeData', () => {
  const schemas: ERC725JSONSchema[] = [
    {
      name: 'LSP3IssuedAssets[]',
      key: '0x3a47ab5bd3a594c3a8995f8fa58d0876c96819ca4516bd76100c92462f2f9dc0',
      keyType: 'Array',
      valueContent: 'Address',
      valueType: 'address',
    },
    {
      name: 'LSP1UniversalReceiverDelegate',
      key: '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
```

```
    keyType: 'Singleton',
    valueType: 'address',
    valueContent: 'Address',
  },

  {
    name: 'LSP3Profile',
    key: '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
    keyType: 'Singleton',
    valueType: 'bytes',
    valueContent: 'JSONURL',
  },
];

const expectedResult = {
  keys: [
    '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
  ],
  values: ['0x1183790f29be3cdfd0a102862fea1a4a30b3adab'],
};

it('encodes data with named key - [array input]', () => {
  const encodedDataByNamedKey = encodeData(
    [
      {
        keyName: 'LSP1UniversalReceiverDelegate',
        value: '0x1183790f29BE3cDfD0A102862fEA1a4a30b3AdAb',
      },
    ],
    schemas,
  );
  assert.deepStrictEqual(encodedDataByNamedKey, expectedResult);
});

it('encodes data with named key - [non array input]', () => {
  const encodedDataByNamedKey = encodeData(
    {
      keyName: 'LSP1UniversalReceiverDelegate',
      value: '0x1183790f29BE3cDfD0A102862fEA1a4a30b3AdAb',
    },

    schemas,
  );
  assert.deepStrictEqual(encodedDataByNamedKey, expectedResult);
});

it('encodes data with hashed key', () => {
  const hashedKey =
    '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47';

  const encodedDataByHashKey = encodeData(
    [
```

```
      {
        keyName: hashedKey,
        value: '0x1183790f29BE3cDfD0A102862fEA1a4a30b3AdAb',
      },
    ],
    schemas,
  );
  assert.deepStrictEqual(encodedDataByHashKey, expectedResult);
});

it('encodes data with hashed key without 0x prefix', () => {
  const hashedKey =
    '0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47';

  const encodedDataByHashKeyWithout0xPrefix = encodeData(
    [
      {
        keyName: hashedKey,
        value: '0x1183790f29BE3cDfD0A102862fEA1a4a30b3AdAb',
      },
    ],
    schemas,
  );

  assert.deepStrictEqual(
    encodedDataByHashKeyWithout0xPrefix,
    expectedResult,
  );
});

it('encodes array', () => {
  const encodedDataWithMultipleKeys = encodeData(
    [
      {
        keyName: 'LSP3IssuedAssets[]',
        value: ['0xa3e6F38477D45727F6e6f853Cdb479b0D60c0aC9'],
      },
    ],
    schemas,
  );

  assert.deepStrictEqual(encodedDataWithMultipleKeys, {
    keys: [
      '0x3a47ab5bd3a594c3a8995f8fa58d0876c96819ca4516bd76100c92462f2f9dc0',
      '0x3a47ab5bd3a594c3a8995f8fa58d087600000000000000000000000000000000',
    ],
    values: [
      '0x0000000000000000000000000000001',
      '0xa3e6f38477d45727f6e6f853cdb479b0d60c0ac9',
    ],
  });
});
```

```
it('encodes array', () => {
  const addressArray = [
    '0x64132555d24b8fbf81d2d65214c485c694cb3d4b4',
    '0xd6c68c2c94af899ce43ff1863693016a711ae7c7',
    '0x79b698f4bc3051f18b5f94046f09d70823a8fd44',
    '0x72bebf88546525a5888f188b390701bb0fd9b1a5',
    '0x882aca051979e32e787e8815d9880759f91e7124',
    '0x78827c8f8205072858a8cce39b8724d948327ba0',
    '0xe27cd9c132677cdce2e9efa43b040de35ceff069',
    '0x072616745957b45c8989e12b95633900fafac4ebe',
    '0xfd5a7c50c0cf665a772407af3f055227845 9c44',
    '0x13de082cf8a499eee75b0681cfa0141a145f15d9',
    '0xe3610d0eb167fe7a7b7c25d0aee8874eb8b113ef',
  ];
  const encodedDataWithMultipleKeys = encodeData(
    [
      {
        keyName: 'LSP3IssuedAssets[]',
        value: addressArray,
      },
    ],
    schemas,
  );

  assert.deepStrictEqual(encodedDataWithMultipleKeys, {
    keys: [
      '0x3a47ab5bd3a594c3a8995f8fa58d0876c96819ca4516bd76100c92462f2f9dc0',
      '0x3a47ab5bd3a594c3a8995f8fa58d087600000000000000000000000000000000',
      '0x3a47ab5bd3a594c3a8995f8fa58d087600000000000000000000000000000001',
      '0x3a47ab5bd3a594c3a8995f8fa58d087600000000000000000000000000000002',
      '0x3a47ab5bd3a594c3a8995f8fa58d087600000000000000000000000000000003',
      '0x3a47ab5bd3a594c3a8995f8fa58d087600000000000000000000000000000004',
      '0x3a47ab5bd3a594c3a8995f8fa58d087600000000000000000000000000000005',
      '0x3a47ab5bd3a594c3a8995f8fa58d087600000000000000000000000000000006',
      '0x3a47ab5bd3a594c3a8995f8fa58d087600000000000000000000000000000007',
      '0x3a47ab5bd3a594c3a8995f8fa58d087600000000000000000000000000000008',
      '0x3a47ab5bd3a594c3a8995f8fa58d087600000000000000000000000000000009',
      '0x3a47ab5bd3a594c3a8995f8fa58d08760000000000000000000000000000000a',
    ],
    values: ['0x000000000000000000000000000000000b', ...addressArray],
  });
});

it('encodes multiple keys', () => {
  const encodedMultipleKeys = encodeData(
    [
      {
        keyName: 'LSP3Profile',
        value: {
          verification: {
            method: 'keccak256(utf8)',
```

```
            data: '0x820464ddfac1bec070cc14a8daf04129871d458f2ca94368aae8391311af6361',
          },
          url: 'ipfs://QmYr1VJLwerg6pEoscdhVGugo39pa6rycEZLjtRPDfW84UAx',
        },
      },
      {
        keyName: 'LSP3IssuedAssets[]',
        value: [
          '0xD94353D9B005B3c0A9Da169b768a31C57844e490',
          '0xDaea594E385Fc724449E3118B2Db7E86dFBa1826',
        ],
      },
      {
        keyName: 'LSP1UniversalReceiverDelegate',
        value: '0x1183790f29BE3cDfD0A102862fEA1a4a30b3AdAb',
      },
    ],
    schemas,
  );

  assert.deepStrictEqual(encodedMultipleKeys, {
    keys: [
      '0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
      '0x3a47ab5bd3a594c3a8995f8fa58d0876c96819ca4516bd76100c92462f2f9dc0',
      '0x3a47ab5bd3a594c3a8995f8fa58d0876000000000000000000000000000000000000',
      '0x3a47ab5bd3a594c3a8995f8fa58d0876000000000000000000000000000000000001',
      '0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47',
    ],
    values: [

'0x6f357c6a820464ddfac1bec070cc14a8daf04129871d458f2ca94368aae8391311af6361697066733a2f2f516d597231564a4c776572673670456f73636468564775676f333970613672796345a4c6a745250446657383455417 8',
      '0x0000000000000000000000000000002',
      '0xd94353d9b005b3c0a9da169b768a31c57844e490',
      '0xdaea594e385fc724449e3118b2db7e86dfba1826',
      '0x1183790f29be3cdfd0a102862fea1a4a30b3adab',
    ],
  });
});

it('encodes dynamic keys', () => {
  const address = '0x78c964cd805233eb39f2db152340079088809725';

  const encodedDynamicKeys = encodeData(
    [
      {
        keyName: 'DynamicKey:<address>',
        dynamicKeyParts: [address],
        value: '0xc57390642767fc9adb0e4211fac735abe2edcfde',
      },
      {
```

```
        keyName: 'DynamicKey:<bytes4>:<string>',
        dynamicKeyParts: ['0x11223344', 'Summer'],
        value: '0x5bed9e061cea8b4be17d3b5ea85de62f483a40fd',
      },
    ],
    [
      {
        name: 'DynamicKey:<address>',
        key: '0x0fb367364e1852abc5f20000<address>',
        keyType: 'Mapping',
        valueType: 'bytes',
        valueContent: 'Address',
      },
      {
        name: 'DynamicKey:<bytes4>:<string>',
        key: '0xForDynamicKeysThisFieldIsIrrelevantAndWillBeOverwriten',
        keyType: 'Mapping',
        valueType: 'bytes',
        valueContent: 'Address',
      },
    ],
  );

  assert.deepStrictEqual(encodedDynamicKeys, {
    keys: [
      &amp;nbsp;0x0fb367364e1852abc5f20000${address.replace('0x', '')}&amp;nbsp;,
      '0x0fb367364e181122334400007746e4c8ba6f946d9f51a1c9e539fb62598962aa',
    ],
    values: [
      '0xc57390642767fc9adb0e4211fac735abe2edcfde',
      '0x5bed9e061cea8b4be17d3b5ea85de62f483a40fd',
    ],
  });
 });
});

describe('isDataAuthentic', () => {
 it('returns true if data is authentic', () => {
  const data = 'h3ll0HowAreYou?';
  const expectedHash = keccak256(data);

  const isAuthentic = isDataAuthentic(data, {
    data: expectedHash,
    method: SUPPORTED_VERIFICATION_METHOD_STRINGS.KECCAK256_BYTES,
  });

  assert.ok(isAuthentic);
 });
 it('returns false if data is not authentic', () => {
  const data = 'h3ll0HowAreYou?';
  const expectedHash = 'wrongHash';
```

```
    const isAuthentic = isDataAuthentic(data, {
      data: expectedHash,
      method: SUPPORTED_VERIFICATION_METHOD_STRINGS.KECCAK256_BYTES,
    });

    assert.strictEqual(isAuthentic, false);
  });
});

describe('guessKeyTypeFromKeyName', () => {
  const testCases: { keyType: ERC725JSONSchemaKeyType; keyName: string }[] = [
    {
      keyType: 'Singleton',
      keyName: 'MyKeyName',
    },
    {
      keyType: 'Array',
      keyName: 'LSP3IssuedAssets[]',
    },
    {
      keyType: 'Mapping',
      keyName: 'SupportedStandards:LSP3Profile',
    },
    {
      keyType: 'Mapping',
      keyName: 'MyCoolAddress:0xcafecafecafecafecafecafecafecafecafecafe',
    },
    {
      keyType: 'Mapping',
      keyName: 'MyCoolAddress:cafecafecafecafecafecafecafecafecafecafe',
    },
    {
      keyType: 'MappingWithGrouping',
      keyName:
        'AddressPermissions:Permissions:cafecafecafecafecafecafecafecafecafecafe',
    },
    {
      keyType: 'MappingWithGrouping',
      keyName:
        'AddressPermissions:Permissions:0xcafecafecafecafecafecafecafecafecafecafe',
    },
  ];

  testCases.forEach((testCase) => {
    it( guesses ${testCase.keyType} , () => {
      assert.deepStrictEqual(
        guessKeyTypeFromKeyName(testCase.keyName),
        testCase.keyType,
      );
    });
  });
});
```

```
describe('convertIPFSGatewayUrl', () => {
  const expectedIPFSGateway = 'https://cloudflare-ipfs.com/ipfs/';

  it('converts when missing /ipfs/', () => {
    assert.deepStrictEqual(
      convertIPFSGatewayUrl('https://cloudflare-ipfs.com'),
      expectedIPFSGateway,
    );
  });
  it('converts when missing /', () => {
    assert.deepStrictEqual(
      convertIPFSGatewayUrl('https://cloudflare-ipfs.com/ipfs'),
      expectedIPFSGateway,
    );
  });
  it('converts when missing ipfs/', () => {
    assert.deepStrictEqual(
      convertIPFSGatewayUrl('https://cloudflare-ipfs.com/'),
      expectedIPFSGateway,
    );
  });
  it('does not convert when passed correctly', () => {
    assert.deepStrictEqual(
      convertIPFSGatewayUrl('https://cloudflare-ipfs.com/ipfs/'),
      expectedIPFSGateway,
    );
  });
});

describe('generateSchemasFromDynamicKeys', () => {
  it('generates a non dynamic schema correctly', () => {
    const schemas: ERC725JSONSchema[] = [
      {
        name: 'AddressPermissions:AllowedFunctions:<address>',
        key: '0x4b80742de2bf8efea1e80000<address>',
        keyType: 'MappingWithGrouping',
        valueType: 'bytes4[]',
        valueContent: 'Bytes4',
      },
      {
        name: 'AddressPermissions[]',
        key: '0xdf30dba06db6a30e65354d9a64c609861f089545ca58c6b4dbe31a5f338cb0e3',
        keyType: 'Array',
        valueType: 'address',
        valueContent: 'Address',
      },
      {
        name: 'LSP4CreatorsMap:<address>',
        key: '0x6de85eaf5d982b4e5da00000<address>',
        keyType: 'Mapping',
        valueType: 'bytes',
```

```
        valueContent: 'Bytes4',
      },
    ];

    const keys: Array<string | GetDataDynamicKey> = [
      'AddressPermissions[]',
      {
        keyName: 'LSP4CreatorsMap:<address>',
        dynamicKeyParts: '0xcafecafecafecafecafecafecafecafecafecafe',
      },
    ];

    const generatedSchemas = generateSchemasFromDynamicKeys(keys, schemas);

    expect(generatedSchemas.length).to.equal(keys.length);

    generatedSchemas.forEach((schema) => {
      expect(
        isDynamicKeyName(schema.name),
        'generated schema key should not be dynamic',
      ).to.be.false;
    });
  });
 });
});
```

</file>

<file>

path: /src/lib/utils.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/lib/utils.ts

```
/*
    This file is part of @erc725/erc725.js.
    @erc725/erc725.js is free software: you can redistribute it and/or modify
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    @erc725/erc725.js is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.
    You should have received a copy of the GNU Lesser General Public License
    along with web3.js.  If not, see <http://www.gnu.org/licenses/>.
*/
/**
 * @file lib/utils.ts
 * @author Robert McLeod <@robertdavid010>
 * @author Fabian Vogelsteller <fabian@lukso.network>
```

```
 * @author Hugo Masclet <@Hugoo>
 * @date 2020
 */

import {
  checkAddressChecksum,
  isAddress,
  numberToHex,
  padLeft,
  stripHexPrefix,
} from 'web3-utils';
import { arrToBufArr } from 'ethereumjs-util';

import {
  JSONURLDataToEncode,
  EncodeDataReturn,
  URLDataWithHash,
  Verification,
} from '../types';
import {
  ERC725JSONSchema,
  ERC725JSONSchemaKeyType,
  ERC725JSONSchemaValueType,
} from '../types/ERC725JSONSchema';

import {
  HASH_METHODS,
  SUPPORTED_VERIFICATION_METHODS,
  SUPPORTED_VERIFICATION_METHODS_LIST,
  COMPACT_BYTES_ARRAY_STRING,
} from '../constants/constants';
import {
  decodeValueContent,
  decodeValueType,
  encodeValueContent,
  encodeValueType,
  valueContentEncodingMap as valueContentMap,
} from './encoder';
import { AssetURLEncode } from '../types/encodeData';
import { isDynamicKeyName } from './encodeKeyName';
import { getSchemaElement } from './getSchemaElement';
import { EncodeDataInput } from '../types/decodeData';
import { GetDataDynamicKey } from '../types/GetData';
import { isValidTuple } from './decodeData';

/**
 *
 * @param {string} valueContent as per ERC725Schema definition
 * @param {string} valueType as per ERC725Schema definition
 * @param decodedValue can contain single value, an array, or an object as required by schema
(JSONURL, or ASSETURL)
 * @param {string} [name]
```

```
 *
 * @return the encoded value as per the schema
 */
export function encodeKeyValue(
  valueContent: string,
  valueType: ERC725JSONSchemaValueType,
  decodedValue:
    | string
    | string[]
    | number
    | number[]
    | JSONURLDataToEncode
    | JSONURLDataToEncode[]
    | boolean,
  name?: string,
): string | false {
  const isSupportedValueContent =
    !!valueContentMap(valueContent) || valueContent.slice(0, 2) === '0x';

  if (!isSupportedValueContent) {
    throw new Error(
       The valueContent '${valueContent}'
          for ${name} is not supported. ,
    );
  }

  const isValueTypeArray = valueType.slice(valueType.length - 2) === '[]';

  if (!isValueTypeArray && !Array.isArray(decodedValue)) {
    // Straight forward encode
    return encodeValueContent(valueContent, decodedValue);
  }

  const valueContentEncodingMethods = valueContentMap(valueContent);

  const isSameEncoding =
    valueContentEncodingMethods &&
    valueContentEncodingMethods.type === valueType.split('[]')[0];

  let result;

  // We only loop if the valueType done by abi.encodeParameter can not handle it directly
  if (Array.isArray(decodedValue)) {
    // value type encoding will handle it?

    // we handle an array element encoding
    const results: Array<string | AssetURLEncode | false> = [];
    for (let index = 0; index < decodedValue.length; index++) {
      const element = decodedValue[index];
      results.push(encodeValueContent(valueContent, element));
    }
```

```
    result = results;
  }

  if (
    // and we only skip bytes regardless
    valueType !== 'bytes' &&
    // Requires encoding because !sameEncoding means both encodings are required
    !isSameEncoding
  ) {
    result = encodeValueType(valueType, result);
  } else if (isValueTypeArray && isSameEncoding) {
    result = encodeValueType(valueType, decodedValue as any);
  }

  return result;
}

/**
 *
 * @param key The schema key of a schema with keyType = 'Array'
 * @param index An integer representing the intended array index
 * @return The raw bytes key for the array element
 */
export function encodeArrayKey(key: string, index: number) {
  return key.slice(0, 34) + padLeft(numberToHex(index), 32).replace('0x', '');
}

/**
 *
 * @param keyName the schema key name
 * @returns a guess of the schema key type
 */
export function guessKeyTypeFromKeyName(
  keyName: string,
): ERC725JSONSchemaKeyType {
  // This function could not work with subsequents keys of an Array
  // It will always assume the given key, if array, is the initial array key.

  const splittedKeyName = keyName.split(':');

  if (splittedKeyName.length === 3) {
    return 'MappingWithGrouping';
  }

  if (splittedKeyName.length === 2) {
    return 'Mapping';
  }

  if (keyName.substring(keyName.length - 2, keyName.length) === '[]') {
    return 'Array';
  }
```

```
  return 'Singleton';
}

export const encodeTupleKeyValue = (
  valueContent: string, // i.e. (bytes4,Number,bytes16)
  valueType: string, // i.e. (bytes4,bytes8,bytes16)
  decodedValues: Array<string | number | JSONURLDataToEncode | string[]>,
) => {
  // We assume data has already been validated at this stage

  const valueTypeParts = valueType
    .substring(1, valueType.length - 1)
    .split(',');
  const valueContentParts = valueContent
    .substring(1, valueContent.length - 1)
    .split(',');

  if (valueTypeParts.length !== decodedValues.length) {
    throw new Error(
       Can not encode tuple key value: ${decodedValues}. Expecte array of length:
${valueTypeParts.length} ,
    );
  }

  const returnValue =
     0x  +
    valueContentParts
      .map((valueContentPart, i) => {
        const encodedKeyValue = encodeKeyValue(
          valueContentPart,
          valueTypeParts[i],
          decodedValues[i],
        );

        if (!encodedKeyValue) {
          return ''; // may cause issues?
        }

        const numberOfBytes = parseInt(valueTypeParts[i].substring(5), 10); // bytes50 -> 50

        // If the encoded value is too large for the expected valueType, we shrink it from the left
        // i.e. number are encoded on 32bytes
        // TODO: might be missing cases !
        if (encodedKeyValue.length > 2 + numberOfBytes * 2) {
          return encodedKeyValue.slice(
            encodedKeyValue.length - numberOfBytes * 2,
          );
        }

        return padLeft(encodedKeyValue, numberOfBytes * 2).replace('0x', '');
      })
      .join('');
```

```
  return returnValue;
};

/**
 *
 * @param schema is an object of a schema definitions.
 * @param value will be either key-value pairs for a key type of Array, or a single value for type Singleton.
 *
 * @return the encoded value for the key as per the supplied schema.
 */
export function encodeKey(
  schema: ERC725JSONSchema,
  value:
    | string
    | number
    | (string | number)[]
    | string[][]
    | JSONURLDataToEncode
    | JSONURLDataToEncode[]
    | boolean,
) {
  // NOTE: This will not guarantee order of array as on chain. Assumes developer must set correct order

  const lowerCaseKeyType = schema.keyType.toLowerCase();

  switch (lowerCaseKeyType) {
    case 'array': {
      if (!Array.isArray(value)) {
        console.error("Can't encode a non array for key of type array");
        return null;
      }

      const results: { key: string; value: string }[] = [];

      for (let index = 0; index < value.length; index++) {
        const dataElement = value[index];
        if (index === 0) {
          // This is arrayLength as the first element in the raw array
          // encoded as uint128
          results.push({
            key: schema.key,
            value: encodeValueType('uint128', value.length),
          });
        }

        results.push({
          key: encodeArrayKey(schema.key, index),
          value: encodeKeyValue(
            schema.valueContent,
            schema.valueType,
            dataElement,
```

```
        schema.name,
      ) as string,
    });
  }

  return results;
}
case 'mappingwithgrouping':
case 'singleton':
case 'mapping':
  if (isValidTuple(schema.valueType, schema.valueContent)) {
    if (!Array.isArray(value)) {
      throw new Error(
         Incorrect value for tuple. Got: ${value}, expected array. ,
      );
    }

    const isCompactBytesArray: boolean = schema.valueType.includes(
      COMPACT_BYTES_ARRAY_STRING,
    );

    if (Array.isArray(value[0]) && isCompactBytesArray) {
      const valueType = schema.valueType.replace(
        COMPACT_BYTES_ARRAY_STRING,
        '',
      );
      const valueContent = schema.valueContent.replace(
        COMPACT_BYTES_ARRAY_STRING,
        '',
      );

      const encodedTuples = value.map((element) => {
        return encodeTupleKeyValue(valueContent, valueType, element);
      });
      return encodeValueType('bytes[CompactBytesArray]', encodedTuples);
    }

    return encodeTupleKeyValue(
      schema.valueContent,
      schema.valueType,
      value,
    );
  }

  // This adds an extra check to ensure the casting below is safe
  // TODO: refactor to fix the TS typing.
  if (
    Array.isArray(value) &&
    Array.isArray(value[0]) &&
    !isValidTuple(schema.valueType, schema.valueContent)
  ) {
    throw new Error('Incorrect value for nested array: not a tuple.');
```

```
    }

    return encodeKeyValue(
      schema.valueContent,
      schema.valueType,
      value as
        | string
        | string[]
        | number
        | number[]
        | JSONURLDataToEncode
        | JSONURLDataToEncode[],
      schema.name,
    );
  default:
    console.error(
      'Incorrect data match or keyType in schema from encodeKey(): "' +
        schema.keyType +
        '"',
    );
    return null;
 }
}

/**
 *
 * @param {string} valueContent as per ERC725Schema definition.
 * @param {string} valueType as per ERC725Schema definition.
 * @param {string} value the encoded value as string.
 * @param {string} [name]
 *
 * @return the decoded value as per the schema.
 */
export function decodeKeyValue(
 valueContent: string,
 valueType: ERC725JSONSchemaValueType,
 value,
 name?: string,
) {
 // Check for the missing map.
 const valueContentEncodingMethods = valueContentMap(valueContent);

 if (!valueContentEncodingMethods && valueContent.slice(0, 2) !== '0x') {
  throw new Error(
    'The valueContent "' +
      valueContent +
      '" for "' +
      name +
      '" is not supported.',
  );
 }
```

```
  let sameEncoding =
    valueContentEncodingMethods &&
    valueContentEncodingMethods.type === valueType.split('[]')[0];
  const isArray = valueType.substring(valueType.length - 2) === '[]';

  // VALUE TYPE
  const valueTypeIsBytesNonArray =
    valueType.slice(0, 5) === 'bytes' && valueType.slice(-2) !== '[]';

  if (
    !valueTypeIsBytesNonArray &&
    valueType !== 'string' &&
    !isAddress(value) // checks for addresses, since technically an address is bytes?
  ) {
    // eslint-disable-next-line no-param-reassign
    value = decodeValueType(valueType, value);
  }

  // As per exception above, if address and sameEncoding, then the address still needs to be handled
  if (sameEncoding && isAddress(value) && !checkAddressChecksum(value)) {
    sameEncoding = !sameEncoding;
  }

  if (sameEncoding && valueType !== 'string') {
    return value;
  }

  // VALUE CONTENT
  // We are finished if duplicated encoding methods

  if (isArray && Array.isArray(value)) {
    // value must be an array also
    const results: (string | URLDataWithHash | number | null | boolean)[] = [];

    for (let index = 0; index < value.length; index++) {
      const element = value[index];
      results.push(decodeValueContent(valueContent, element));
    }

    return results;
  }

  return decodeValueContent(valueContent, value);
}

/**
 * @param schema an array of schema definitions as per ${@link ERC725JSONSchema}
 * @param data an object of key-value pairs
 */
export function encodeData(
  data: EncodeDataInput | EncodeDataInput[],
  schema: ERC725JSONSchema[],
```

```
): EncodeDataReturn {
  const dataAsArray = Array.isArray(data) ? data : [data];

  return dataAsArray.reduce(
    (accumulator, { keyName, value, dynamicKeyParts }) => {
      let schemaElement: ERC725JSONSchema | null = null;
      let encodedValue; // would be nice to type this

      // Switch between non dynamic and dynamic keys:
      if (isDynamicKeyName(keyName)) {
        // In case of a dynamic key, we need to check if the value is of type DynamicKeyPartIntput.
        if (!dynamicKeyParts) {
          throw new Error(
             Can't encodeData for dynamic key: ${keyName} with non dynamic values. Got: ${value},
expected object. ,
          );
        }

        schemaElement = getSchemaElement(schema, keyName, dynamicKeyParts);
        encodedValue = encodeKey(schemaElement, value);
      } else {
        schemaElement = getSchemaElement(schema, keyName);
        encodedValue = encodeKey(schemaElement, value as any);
      }

      if (typeof encodedValue === 'string') {
        accumulator.keys.push(schemaElement.key);
        accumulator.values.push(encodedValue);
      } else if (encodedValue !== false && encodedValue !== null) {
        encodedValue.forEach((keyValuePair) => {
          accumulator.keys.push(keyValuePair.key);
          accumulator.values.push(keyValuePair.value);
        });
      }

      return accumulator;
    },
    { keys: [], values: [] } as EncodeDataReturn,
  );
}

export function getVerificationMethod(nameOrSig: string) {
  const verificationMethod = Object.values(HASH_METHODS).find(
    ({ name, sig }) => name === nameOrSig || sig === nameOrSig,
  );

  if (!verificationMethod) {
    throw new Error(
       Chosen verification method '${nameOrSig}' is not supported. Supported verification methods:
${SUPPORTED_VERIFICATION_METHODS_LIST} ,
    );
  }
```

```
  return verificationMethod;
}

export function hashData(
  data: string | Uint8Array | Record<string, any>,
  nameOrSig: SUPPORTED_VERIFICATION_METHODS | string,
): string {
  return getVerificationMethod(nameOrSig).method(data);
}

/**
 * Hashes the data received with the specified hashing function,
 * and compares the result with the provided hash.
 */
export function isDataAuthentic(
  data: string | Uint8Array,
  options: Verification,
): boolean {
  let dataHash: string;

  if (!options || !options.method) {
    return true;
  }

  if (data instanceof Uint8Array) {
    dataHash = hashData(arrToBufArr(data), options.method);
  } else {
    dataHash = hashData(data, options.method);
  }

  if (dataHash !== options.data) {
    console.error(
       Hash mismatch, returned JSON hash ("${dataHash}") is different from expected hash:
"${options.method}" ,
    );
    return false;
  }

  return true;
}

/**
 * Transforms passed ipfsGateway url to correct format for fetching IPFS data
 *
 * @param ipfsGateway
 * @return {*}  string converted IPFS gateway URL
 */
export function convertIPFSGatewayUrl(ipfsGateway: string) {
  let convertedIPFSGateway = ipfsGateway;

  if (ipfsGateway.endsWith('/') && !ipfsGateway.endsWith('/ipfs/')) {
```

```
    convertedIPFSGateway = ipfsGateway + 'ipfs/';
  } else if (ipfsGateway.endsWith('/ipfs')) {
    convertedIPFSGateway = ipfsGateway + '/';
  } else if (!ipfsGateway.endsWith('/ipfs/')) {
    convertedIPFSGateway = ipfsGateway + '/ipfs/';
  }

  return convertedIPFSGateway;
}

/**
 * Given a list of keys (dynamic or not) and a list of schemas with dynamic keys, it will
 * generate a "final"/non dynamic schemas list.
 */
export const generateSchemasFromDynamicKeys = (
  keyNames: Array<string | GetDataDynamicKey>,
  schemas: ERC725JSONSchema[],
) => {
  return keyNames.map((keyName) => {
    if (typeof keyName === 'string') {
      return getSchemaElement(schemas, keyName);
    }
    return getSchemaElement(schemas, keyName.keyName, keyName.dynamicKeyParts);
  });
};

/**
 * Changes the protocol from  ipfs://  to  http(s)://  and adds the selected IPFS
gateway.
 *  ipfs://QmbKvCVEePiDKxuouyty9bMsWBAxZDGr2jhxd4pLGLx95D =>
https://ipfs.lukso.network/ipfs/QmbKvCVEePiDKxuouyty9bMsWBAxZDGr2jhxd4pLGLx95D 
 */
export function patchIPFSUrlsIfApplicable(
  receivedData: URLDataWithHash,
  ipfsGateway: string,
): URLDataWithHash {
  if (
    receivedData &&
    receivedData.url &&
    receivedData.url.indexOf('ipfs://') !== -1
  ) {
    return {
      ...receivedData,
      url: receivedData.url.replace('ipfs://', ipfsGateway),
    };
  }

  return receivedData;
}

export function countNumberOfBytes(data: string) {
  return stripHexPrefix(data).length / 2;
```

```
}
```

</file>

<file>

path: /src/provider/providerWrapper.test.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/provider/providerWrapper.test.ts

```
import assert from 'assert';

import { ProviderWrapper } from './providerWrapper';

const erc725AccountAddress = '0x214be121bB52e6909c5158579b3458f8760f1b2f';
const defaultGas = 1_000_000;

describe('ProviderWrapper', () => {
  describe('#getOwner', () => {
    it('should return an address', async () => {
      const mockProvider = {
        send: (_payload, cb) => {
          cb(null, {
            result:
              '0x000000000000000000000000a78e0e7c9b1b36f7e25c5ccdfdba005ec37eadf4',
          });
        },
      };
      const ethSource = new ProviderWrapper(mockProvider, defaultGas);

      const owner = await ethSource.getOwner(erc725AccountAddress);
      assert.deepStrictEqual(
        owner,
        '0xA78E0E7C9b1B36F7E25C5CcDfdbA005Ec37eadf4',
      );
    });

    it('should throw when promise was rejected', async () => {
      const mockProvider = {
        send: (_payload, cb) => {
          cb(new Error('some error'));
        },
      };
      const ethSource = new ProviderWrapper(mockProvider, defaultGas);

      try {
        await ethSource.getOwner(erc725AccountAddress);
      } catch (error: any) {
        assert.deepStrictEqual(error.message, 'some error');
      }
    });
```

```
  it('should throw when promise returned error', async () => {
    const mockProvider = {
      send: (_payload, cb) => {
        cb(null, {
          error: new Error('some error'),
        });
      },
    };
    const ethSource = new ProviderWrapper(mockProvider, defaultGas);

    try {
      await ethSource.getOwner(erc725AccountAddress);
    } catch (error: any) {
      assert.deepStrictEqual(error.message, 'some error');
    }
  });
 });
});
```

</file>

<file>

path: /src/provider/providerWrapper.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/provider/providerWrapper.ts

```
/*
    This file is part of @erc725/erc725.js.
    @erc725/erc725.js is free software: you can redistribute it and/or modify
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    @erc725/erc725.js is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.
    You should have received a copy of the GNU Lesser General Public License
    along with web3.js.  If not, see <http://www.gnu.org/licenses/>.
*/
/**
 * @file providers/web3ProviderWrapper.ts
 * @author Robert McLeod <@robertdavid010>, Fabian Vogelsteller <fabian@lukso.network>
 * @date 2020
 */

/*
  This file will handle querying the Ethereum web3 rpc based on a given provider
  in accordance with implementation of smart contract interfaces of ERC725
*/
```

```
import AbiCoder from 'web3-eth-abi';

import { JsonRpc } from '../types/JsonRpc';
import { Method } from '../types/Method';
import { constructJSONRPC, decodeResult } from '../lib/provider-wrapper-utils';
import { ProviderTypes } from '../types/provider';
import { ERC725_VERSION, ERC725Y_INTERFACE_IDS } from '../constants/constants';

const abiCoder = AbiCoder;

interface GetDataReturn {
  key: string;
  value: Record<string, any> | null;
}

export class ProviderWrapper {
  type: ProviderTypes;
  provider: any;
  gas: number;
  constructor(provider: any, gasInfo: number) {
    if (typeof provider.request === 'function') {
      this.type = ProviderTypes.ETHEREUM;
    } else {
      this.type = ProviderTypes.WEB3;
    }
    this.provider = provider;
    this.gas = gasInfo;
  }

  async getOwner(address: string) {
    const result = await this.callContract(
      constructJSONRPC(address, Method.OWNER, this.gas),
    );
    if (result.error) {
      throw result.error;
    }

    return decodeResult(Method.OWNER, result.result);
  }

  async getErc725YVersion(address: string): Promise<ERC725_VERSION> {
    const isErc725Yv5 = await this.supportsInterface(
      address,
      ERC725Y_INTERFACE_IDS['5.0'],
    );

    if (isErc725Yv5) {
      return ERC725_VERSION.ERC725_v5;
    }

    const isErc725Yv3 = await this.supportsInterface(
```

```
    address,
    ERC725Y_INTERFACE_IDS['3.0'],
  );

  // The version 3 of the package can use the getData function from v2, still compatible
  if (isErc725Yv3) {
    return ERC725_VERSION.ERC725_v2;
  }

  const isErc725Yv2 = await this.supportsInterface(
    address,
    ERC725Y_INTERFACE_IDS['2.0'],
  );

  if (isErc725Yv2) {
    return ERC725_VERSION.ERC725_v2;
  }

  // v0.2.0 and v0.6.0 have the same function signatures for getData, only versions before v0.2.0 requires a
different call

  const isErc725YLegacy = await this.supportsInterface(
    address,
    ERC725Y_INTERFACE_IDS.legacy,
  );

  return isErc725YLegacy
    ? ERC725_VERSION.ERC725_LEGACY
    : ERC725_VERSION.NOT_ERC725;
}

/**
 * https://eips.ethereum.org/EIPS/eip-165
 *
 * @param address the smart contract address
 * @param interfaceId ERC-165 identifier as described here:
https://github.com/ERC725Alliance/ERC725/blob/develop/docs/ERC-725.md#specification
 */
async supportsInterface(
  address: string,
  interfaceId: string,
): Promise<boolean> {
  const result = await this.callContract(
    constructJSONRPC(
      address,
      Method.SUPPORTS_INTERFACE,
      this.gas,
       ${interfaceId}${'0000000000000000000000000000000000000000000000000000000000000000'} ,
    ),
  );

  // These will be boolean because passing Method.SUPPORTS_INTERFACE ensures they will be
```

```
decoded to bool by web3-eth-abi lib
    // The {[key: string]: any} return type causes problems for boolean values so we have to cast here
    if (this.type === ProviderTypes.ETHEREUM) {
      return decodeResult(
        Method.SUPPORTS_INTERFACE,
        result,
      ) as unknown as boolean;
    }
    return decodeResult(
      Method.SUPPORTS_INTERFACE,
      result.result,
    ) as unknown as boolean;
  }

  /**
   * https://eips.ethereum.org/EIPS/eip-1271
   *
   * @param address the contract address
   * @param hash
   * @param signature
   */
  async isValidSignature(
    address: string,
    hash: string,
    signature: string,
  ): Promise<string> {
    if (this.type === ProviderTypes.ETHEREUM) {
      const encodedParams = abiCoder.encodeParameters(
        ['bytes32', 'bytes'],
        [hash, signature],
      );

      const result = await this.callContract(
        constructJSONRPC(
          address,
          Method.IS_VALID_SIGNATURE,
          this.gas,
          encodedParams,
        ),
      );

      if (result.error) {
        throw result.error;
      }

      // Passing Method.IS_VALID_SIGNATURE ensures this will be string
      return decodeResult(
        Method.IS_VALID_SIGNATURE,
        result,
      ) as unknown as string;
    }
```

```
  const encodedParams = abiCoder.encodeParameters(
    ['bytes32', 'bytes'],
    [hash, signature],
  );

  const results = await this.callContract([
    constructJSONRPC(
      address,
      Method.IS_VALID_SIGNATURE,
      this.gas,
      encodedParams,
    ),
  ]);
  if (results.error) {
    throw results.error;
  }

  // Passing Method.IS_VALID_SIGNATURE ensures this will be string
  return decodeResult(
    Method.IS_VALID_SIGNATURE,
    results[0].result,
  ) as unknown as string;
}

async getData(address: string, keyHash: string) {
  const result = await this.getAllData(address, [keyHash]);
  try {
    return result[0].value;
  } catch {
    return null;
  }
}

async getAllData(
  address: string,
  keyHashes: string[],
): Promise<GetDataReturn[]> {
  const erc725Version = await this.getErc725YVersion(address);

  if (erc725Version === ERC725_VERSION.NOT_ERC725) {
    throw new Error(
       Contract: ${address} does not support ERC725Y interface. ,
    );
  }

  switch (erc725Version) {
    case ERC725_VERSION.ERC725_v5:
      return this._getAllDataGeneric(
        address,
        keyHashes,
        Method.GET_DATA_BATCH,
      );
```

```
    case ERC725_VERSION.ERC725_v2:
      return this._getAllDataGeneric(address, keyHashes, Method.GET_DATA);
    case ERC725_VERSION.ERC725_LEGACY:
      return this._getAllDataLegacy(address, keyHashes);
    default:
      return [];
  }
}

private async _getAllDataGeneric(
  address: string,
  keyHashes: string[],
  method: Method.GET_DATA | Method.GET_DATA_BATCH,
): Promise<GetDataReturn[]> {
  if (this.type === ProviderTypes.ETHEREUM) {
    const encodedResults = await this.callContract(
      constructJSONRPC(
        address,
        method,
        this.gas,
        abiCoder.encodeParameter('bytes32[]', keyHashes),
      ),
    );

    const decodedValues = decodeResult(method, encodedResults);

    return keyHashes.map<GetDataReturn>((keyHash, index) => ({
      key: keyHash,
      value: decodedValues ? decodedValues[index] : decodedValues,
    }));
  }

  const payload: JsonRpc[] = [
    constructJSONRPC(
      address,
      method,
      this.gas,
      abiCoder.encodeParameter('bytes32[]', keyHashes),
    ),
  ];

  const results: any = await this.callContract(payload);
  const decodedValues = decodeResult(method, results[0].result);

  return keyHashes.map<GetDataReturn>((key, index) => ({
    key,
    value: decodedValues ? decodedValues[index] : decodedValues,
  }));
}

private async _getAllDataLegacy(
  address: string,
```

```
  keyHashes: string[],
 ): Promise<GetDataReturn[]> {
  if (this.type === ProviderTypes.ETHEREUM) {
   // Here we could use &amp;nbsp;getDataMultiple&amp;nbsp; instead of sending multiple calls to
&amp;nbsp;getData&amp;nbsp;
   // But this is already legacy and it won't be used anymore..
   const encodedResultsPromises = keyHashes.map((keyHash) =>
    this.callContract(
     constructJSONRPC(address, Method.GET_DATA_LEGACY, this.gas, keyHash),
    ),
   );

   const decodedResults = await Promise.all(encodedResultsPromises);

   return decodedResults.map((decodedResult, index) => ({
    key: keyHashes[index],
    value: decodeResult(Method.GET_DATA_LEGACY, decodedResult),
   }));
  }

  const payload: JsonRpc[] = [];
  // Here we could use &amp;nbsp;getDataMultiple&amp;nbsp; instead of sending multiple calls to
&amp;nbsp;getData&amp;nbsp;
  // But this is already legacy and it won't be used anymore..
  for (let index = 0; index < keyHashes.length; index++) {
   payload.push(
    constructJSONRPC(
     address,
     Method.GET_DATA_LEGACY,
     this.gas,
     keyHashes[index],
    ),
   );
  }

  const results: any = await this.callContract(payload);

  return payload.map<GetDataReturn>((payloadCall, index) => ({
   key: keyHashes[index],
   value: decodeResult(
    Method.GET_DATA_LEGACY,
    results.find((element) => payloadCall.id === element.id).result,
   ),
  }));
 }

 private async callContract(payload: JsonRpc[] | JsonRpc): Promise<any> {
  if (this.type === ProviderTypes.ETHEREUM) {
   return this.provider.request({
    method: 'eth_call',
    params: (payload as JsonRpc).params,
   });
```

```
    }

    return new Promise((resolve, reject) => {
      // Send old web3 method with callback to resolve promise
      // This is deprecated: https://docs.metamask.io/guide/ethereum-provider.html#ethereum-send-deprecated

      this.provider.send(payload, (e, r) => {
        if (e) {
          reject(e);
        } else {
          resolve(r);
        }
      });
    });
  }
}
```

</file>

<file>

path: /src/schemas/index.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/schemas/index.ts

```
import { ERC725JSONSchema } from '../types/ERC725JSONSchema';

import LSP1UniversalReceiverDelegate from '../../schemas/LSP1UniversalReceiverDelegate.json';
import LSP3Profile from '../../schemas/LSP3ProfileMetadata.json';
import LSP4DigitalAssetLegacy from '../../schemas/LSP4DigitalAssetLegacy.json';
import LSP4DigitalAsset from '../../schemas/LSP4DigitalAsset.json';
import LSP5ReceivedAssets from '../../schemas/LSP5ReceivedAssets.json';
import LSP6KeyManager from '../../schemas/LSP6KeyManager.json';

export default LSP1UniversalReceiverDelegate.concat(
  LSP3Profile,
  LSP4DigitalAssetLegacy,
  LSP4DigitalAsset,
  LSP5ReceivedAssets,
  LSP6KeyManager,
) as ERC725JSONSchema[];
```

</file>

<file>

path: /src/types/Config.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/types/Config.ts

```
import { ERC725JSONSchema } from './ERC725JSONSchema';

export interface ERC725Config {
  /**
   * &amp;nbsp;&amp;nbsp;&amp;nbsp;js title=Example
   * const config = {
   *   ipfsGateway: 'https://ipfs.lukso.network/ipfs/'
   *   gas: 20_000_000 // Optional, default 0
   * };
   * &amp;nbsp;&amp;nbsp;&amp;nbsp;
   * Make sure to use the following format: &amp;nbsp;<url>/ipfs/&amp;nbsp;.<br/>
   * Another example: &amp;nbsp;https://cloudflare-ipfs.com/ipfs/&amp;nbsp;
   */
  ipfsGateway?: string;
  gas?: number;
}

export interface ERC725Options {
  schemas: ERC725JSONSchema[];
  address?: string;
  provider?;
  ipfsGateway: string;
  gas: number;
}
```

</file>

<file>

path: /src/types/ERC725JSONSchema.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/types/ERC725JSONSchema.ts

```
// As Defined in: https://github.com/lukso-network/LIPs/blob/master/LSPs/LSP-2-ERC725YJSONSchema.md

export type ERC725JSONSchemaKeyType =
  | 'Singleton'
  | 'Array'
  | 'Mapping'
  | 'MappingWithGrouping';

export type ERC725JSONSchemaValueContent =
  | 'Number'
  | 'String'
  | 'Address'
  | 'Keccak256'
  | 'AssetURL'
  | 'JSONURL'
  | 'URL'
  | 'Markdown'
  | 'Boolean'
```

```
  | string; // for tuples

export type ERC725JSONSchemaValueType =
  | 'bool'
  | 'boolean'
  | 'string'
  | 'address'
  | 'uint256'
  | 'bytes32'
  | 'bytes'
  | 'bytes4'
  | 'string[]'
  | 'address[]'
  | 'uint256[]'
  | 'bytes32[]'
  | 'bytes4[]'
  | 'bytes[]'
  | 'bool[]'
  | 'boolean[]'
  | string; // for tuples;

/**
 *    javascript title=Example
 *   {
 *     name: "LSP3Profile",
 *     key: "0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5",
 *     keyType: "Singleton",
 *     valueContent: "JSONURL",
 *     valueType: "bytes",
 *   },
 *    
 * Detailed information available on [LSP-2-ERC725YJSONSchema](https://github.com/lukso-network/LIPs/blob/master/LSPs/LSP-2-ERC725YJSONSchema.md)
 */
export interface ERC725JSONSchema {
  name: string; // Describes the name of the key, SHOULD compromise of the Standards name + sub type. e.g: LSP2Name
  key: string; // The keccak256 hash of the name. This is the actual key that MUST be retrievable via ERC725Y.getData(bytes32 key)
  keyType: ERC725JSONSchemaKeyType; // Types that determine how the values should be interpreted.
  valueContent: ERC725JSONSchemaValueContent | string; // string holds '0x1345ABCD...' If the value content are specific bytes, than the returned value is expected to equal those bytes.
  valueType: ERC725JSONSchemaValueType;
}
```

</file>

<file>

path: /src/types/GetData.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/types/GetData.ts

```
export interface GetDataDynamicKey {
  keyName: string;
  dynamicKeyParts: string | string[];
}

export type GetDataInput =
  | string
  | GetDataDynamicKey
  | Array<string | GetDataDynamicKey>;
```

</file>

<file>

path: /src/types/JsonRpc.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/types/JsonRpc.ts

```
interface JsonRpcEthereumProviderParams {
  to: string;
  gas: string;
  value: string;
  data;
}

export type JsonRpcEthereumProviderParamsWithLatest = [
  JsonRpcEthereumProviderParams,
  'latest',
];
export interface JsonRpc {
  jsonrpc: '2.0';
  method: 'eth_call';
  params: JsonRpcEthereumProviderParamsWithLatest;
  id: number;
}
```

</file>

<file>

path: /src/types/Method.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/types/Method.ts

```
export enum Method {
  GET_DATA_LEGACY = 'getDataLegacy', // For legacy ERC725 with interface id: 0x2bd57b73 NOTE: I had to add Legacy at the end so the map keys stays unique
  GET_DATA = 'getData', // For latest ERC725 with interface id: 0x5a988c0f
  GET_DATA_BATCH = 'getDataBatch',
```

```
  OWNER = 'owner',
  SUPPORTS_INTERFACE = 'supportsInterface', // https://eips.ethereum.org/EIPS/eip-165
  IS_VALID_SIGNATURE = 'isValidSignature', // https://eips.ethereum.org/EIPS/eip-1271
}

export enum Encoding {
  BYTES = 'bytes',
  BYTES4 = 'bytes4',
  BOOL = 'bool',
  UINT256 = 'uint256',
  BYTES32_ARRAY = 'bytes32[]',
  BYTES_ARRAY = 'bytes[]',
  ADDRESS = 'address',
}

export interface MethodData {
  sig: string;
  value: string;
  returnEncoding: Encoding;
}

export interface Permissions {
  CHANGEOWNER?: boolean;
  ADDCONTROLLER?: boolean;
  EDITPERMISSIONS?: boolean;
  ADDEXTENSIONS?: boolean;
  CHANGEEXTENSIONS?: boolean;
  ADDUNIVERSALRECEIVERDELEGATE?: boolean;
  CHANGEUNIVERSALRECEIVERDELEGATE?: boolean;
  REENTRANCY?: boolean;
  SUPER_TRANSFERVALUE?: boolean;
  TRANSFERVALUE?: boolean;
  SUPER_CALL?: boolean;
  CALL?: boolean;
  SUPER_STATICCALL?: boolean;
  STATICCALL?: boolean;
  SUPER_DELEGATECALL?: boolean;
  DELEGATECALL?: boolean;
  DEPLOY?: boolean;
  SUPER_SETDATA?: boolean;
  SETDATA?: boolean;
  ENCRYPT?: boolean;
  DECRYPT?: boolean;
  SIGN?: boolean;
  EXECUTE_RELAY_CALL?: boolean;
}
```

</file>

<file>

path: /src/types/decodeData.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/types/decodeData.ts

```
import { EncodeDataType, URLDataWithHash } from './encodeData/JSONURL';

export interface DataInput {
  keyName: string; // can be the name or the hex/hash
  value;
  dynamicKeyParts?: string | string[];
}

export interface EncodeDataInput extends DataInput {
  value: EncodeDataType;
}

export interface DecodeDataInput extends DataInput {
  value: string | { key: string; value: string | null }[];
}

export interface DecodeDataOutput {
  value: string | string[] | URLDataWithHash | null;
  name: string;
  key: string;
}

export interface FetchDataOutput {
  value:
    | null
    | string
    | string[]
    | { LSP3Profile: Record<string, any> }
    | Record<string, any>;
  name: string;
  key: string;
}

export interface GetDataExternalSourcesOutput extends DecodeDataOutput {
  value: any;
}
```

</file>

<file>

path: /src/types/dynamicKeys.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/types/dynamicKeys.ts

```
// Put types / interfaces related to dynamic keys here

import { EncodeDataType } from './encodeData/JSONURL';

export type DynamicKeyParts = string | string[];

export interface DynamicKeyPartInput {
  dynamicKeyParts: DynamicKeyParts;
  value: EncodeDataType;
}

export interface DynamicKeyPart {
  type: string;
  value: string | boolean | number;
}
```

</file>

<file>

path: /src/types/encodeData/JSONURL.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/types/encodeData/JSONURL.ts

```
import { SUPPORTED_VERIFICATION_METHODS } from '../../constants/constants';

export interface KeyValuePair {
  key: string;
  value: any;
}

interface URLData {
  url: string;
}

export interface Verification {
  data: string;
  method: SUPPORTED_VERIFICATION_METHODS | string;
  source?: string;
}

export interface URLDataWithHash extends URLData {
  verification: Verification; // | string is to allow use of string directly without importing the enum
  json?: never;
}

export interface URLDataWithJson extends URLData {
  verification?: Verification;
  json: Record<string, any>;
}

export type JSONURLDataToEncode = URLDataWithHash | URLDataWithJson;

export type EncodeDataType = string | string[] | JSONURLDataToEncode | boolean;

export interface EncodeDataReturn {
  keys: string[];
  values: string[];
}
```

</file>

<file>

path: /src/types/encodeData/index.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/types/encodeData/index.ts

```
import { Verification } from './JSONURL';

export interface AssetURLEncode {
  verification: Verification;
  url: string;
}
```

</file>

<file>

path: /src/types/index.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/types/index.ts

```
export { ERC725Config } from './Config';
export { ProviderTypes } from './provider';
export * from './encodeData/JSONURL';
```

</file>

<file>

path: /src/types/provider.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/src/types/provider.ts

```
export const enum ProviderTypes {
  ETHEREUM = 'ETHEREUM',
  WEB3 = 'WEB3',
}
```

</file>

<file>

path: /test/mockProviders.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/test/mockProviders.ts

```
// This file contains the mock providers used for tests

import AbiCoder from 'web3-eth-abi';

import { METHODS } from '../src/constants/constants';
import { Method } from '../src/types/Method';

interface HttpProviderPayload {
  jsonrpc: '2.0';
  method: string;
  params: Array<{
    to: string;
    gas: string;
    gasPrice: string;
    value: string;
    data: string;
  }>;
  id: number;
```

```
}

const abiCoder = AbiCoder;

const IS_VALID_SIGNATURE_RESPONSE = {
  valid: '0x1626ba7e00000000000000000000000000000000000000000000000000000000',
  notValid:
    '0xffffffff00000000000000000000000000000000000000000000000000000000',
};

export class HttpProvider {
  public returnData: { key: string; value: string }[];
  public supportsInterfaces: string[];
  public isValidSignature: boolean;

  constructor(
    props: { returnData: { key: string; value: string }[] },
    supportsInterfaces: string[],
    isValidSignature?: boolean, // to mock isValidSignature call. If set to true, will return magic value.
  ) {
    // clone array
    this.returnData = [...props.returnData];
    this.supportsInterfaces = supportsInterfaces;
    this.isValidSignature = !!isValidSignature;
  }

  send(payload: HttpProviderPayload | HttpProviderPayload[], cb) {
    if (Array.isArray(payload)) {
      const results: {
        jsonrpc: '2.0';
        id: number;
        result?: string;
        error?: { code: number; message: string; data: string };
      }[] = [];
      for (let index = 0; index < payload.length; index++) {
        const methodSignature = payload[index].params[0].data.slice(0, 10);

        switch (methodSignature) {
          case METHODS[Method.SUPPORTS_INTERFACE].sig:
            throw new Error(
              'Mock of support interface not supported in array mode',
            );

          case METHODS[Method.GET_DATA_LEGACY].sig:
            // The legacy method does not support "multi" mode
            {
              const foundResult = this.returnData.find((element) => {
                // get call param (key)
                const keyParam = '0x' + payload[index].params[0].data.slice(10);
                return element.key === keyParam;
              });
```

```
            results.push({
              jsonrpc: '2.0',
              id: payload[index].id,
              result: foundResult ? foundResult.value : '0x',
            });
          }
          break;
        case METHODS[Method.IS_VALID_SIGNATURE].sig: {
          results.push({
            jsonrpc: '2.0',
            id: payload[index].id,
            result: this.isValidSignature
              ? IS_VALID_SIGNATURE_RESPONSE.valid
              : IS_VALID_SIGNATURE_RESPONSE.notValid,
          });
          break;
        }
        case METHODS[Method.GET_DATA_BATCH].sig:
        case METHODS[Method.GET_DATA].sig:
          // The new ERC725Y allows requesting multiple items in one call
          // getData([A]), getData([A, B, C])...
          //
          {
            const requestedKeys = abiCoder.decodeParameter(
              'bytes32[]',
              payload[index].params[0].data.slice(10),
            );

            const decodedResult = requestedKeys.map((requestedKey) => {
              const foundElement = this.returnData.find((element) => {
                return element.key === requestedKey;
              });
              return foundElement
                ? abiCoder.decodeParameter('bytes[]', foundElement.value)[0] // we need to decode the keys as the values provided to the mock are already bytes[] encoded (as it was made for "single item" request mode)
                : '0x';
            });

            results.push({
              jsonrpc: '2.0',
              id: payload[index].id,
              result: abiCoder.encodeParameter('bytes[]', decodedResult),
            });
          }
          break;
        default:
          throw new Error(
            &amp;nbsp;Method signature: ${methodSignature} mock is not supported in HttpProvider mock [array mode]&amp;nbsp;,
          );
      }
    }
```

```
    setTimeout(() => cb(null, results), 10);
  } else {
    let result: string;

    const methodSignature = payload.params[0].data.slice(0, 10);

    switch (methodSignature) {
      case METHODS[Method.SUPPORTS_INTERFACE].sig:
        {
          const requestedInterface = &amp;nbsp;0x${payload.params[0].data.slice(
            10,
            18,
          )}&amp;nbsp;;
          if (this.supportsInterfaces.includes(requestedInterface)) {
            result =
              '0x0000000000000000000000000000000000000000000000000000000000000001';
          } else {
            result =
              '0x0000000000000000000000000000000000000000000000000000000000000000';
          }
        }
        break;
      case METHODS[Method.GET_DATA_LEGACY].sig:
        {
          const keyParam = '0x' + payload.params[0].data.slice(10);
          const foundResult = this.returnData.find((e) => e.key === keyParam);
          result = foundResult ? foundResult.value : '0x';
        }
        break;
      case METHODS[Method.GET_DATA_BATCH].sig:
      case METHODS[Method.GET_DATA].sig:
        {
          const keyParam = '0x' + payload.params[0].data.slice(138);
          const foundResult = this.returnData.find((e) => e.key === keyParam);
          result = foundResult ? foundResult.value : '0x';
        }
        break;
      default:
        throw new Error(
          &amp;nbsp;Method signature: ${methodSignature} mock is not supported in HttpProvider mock [not array
mode]&amp;nbsp;,
        );
    }

    setTimeout(
      () =>
        cb(null, {
          jsonrpc: '2.0',
          result,
        }),
      10,
```

```
    );
  }
 }
}

interface EthereumProviderPayload {
 method: string;
 params: Array<{
  to: string;
  gas: string;
  gasPrice: string;
  value: string;
  data: string;
 }>;
}

export class EthereumProvider {
 public returnData;
 public supportsInterfaces: string[];
 public isValidSignature: boolean;

 constructor(
  props: { returnData: { key: string; value: string }[] },
  supportsInterfaces: string[],
  isValidSignature?: boolean, // to mock isValidSignature call. If set to true, will return magic value.
 ) {
  // Deconstruct to create local copy of array
  this.returnData = [...props.returnData];
  this.supportsInterfaces = supportsInterfaces;
  this.isValidSignature = !!isValidSignature;
 }

 request(payload: EthereumProviderPayload) {
  let result: string;

  const methodSignature = payload.params[0].data.slice(0, 10);

  switch (methodSignature) {
   case METHODS[Method.SUPPORTS_INTERFACE].sig:
    {
     const requestedInterface = &amp;nbsp;0x${payload.params[0].data.slice(
      10,
      18,
     )}&amp;nbsp;;
     if (this.supportsInterfaces.includes(requestedInterface)) {
      result =
       '0x0000000000000000000000000000000000000000000000000000000000000001';
     } else {
      result =
       '0x0000000000000000000000000000000000000000000000000000000000000000';
     }
    }
```

```
      break;
    case METHODS[Method.GET_DATA_LEGACY].sig:
      {
        const keyParam = '0x' + payload.params[0].data.slice(10);

        result = this.returnData.find((e) => e.key === keyParam)?.value;
      }
      break;
    case METHODS[Method.IS_VALID_SIGNATURE].sig:
      result = this.isValidSignature
        ? IS_VALID_SIGNATURE_RESPONSE.valid
        : IS_VALID_SIGNATURE_RESPONSE.notValid;
      break;
    case METHODS[Method.GET_DATA].sig:
      {
        // Duplicated logic with HttpProvider
        const requestedKeys = abiCoder.decodeParameter(
          'bytes32[]',
          payload.params[0].data.slice(10),
        );

        const decodedResult = requestedKeys.map((requestedKey) => {
          const foundElement = this.returnData.find((element) => {
            return element.key === requestedKey;
          });
          return foundElement
            ? abiCoder.decodeParameter('bytes[]', foundElement.value)[0] // we need to decode the keys as
the values provided to the mock are already bytes[] encoded (as it was made for "single item" request mode)
            : '0x';
        });

        result = abiCoder.encodeParameter('bytes[]', decodedResult);
      }
      break;
    default:
      throw new Error(
        &amp;nbsp;Method signature: ${methodSignature} is not supported in EthereumProvider mock&amp;nbsp;,
      );
  }

  return new Promise((resolve) => {
    setTimeout(() => {
      // TODO: Handle reject
      resolve(result);
    }, 50);
  });
 }
}
```

</file>

<file>

path: /test/mockSchema.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/test/mockSchema.ts

```
// Mock schema for tests
// make one schema that tests every single type

import AbiCoder from 'web3-eth-abi';
import { leftPad, utf8ToHex } from 'web3-utils';

import { ERC725JSONSchema } from '../src/types/ERC725JSONSchema';

const abiCoder = AbiCoder;

export const mockSchema: (ERC725JSONSchema & {
  returnRawData?: string | string[];
  returnRawDataArray?; // Used for the array version of getData()
  returnGraphData?;
  dynamicKeyParts?: string | string[];
  expectedResult?;
})[] = [
  // Case 1
  {
    name: 'SupportedStandards:LSP3Profile',
    key: '0xeafec4d89fa9619884b600005ef83ad9559033e6e941db7d7c495acdce616347',
    keyType: 'Mapping',
    valueContent: '0x5ef83ad9',
    valueType: 'bytes',
    // Testing data
    returnRawData: abiCoder.encodeParameter('bytes', '0x5ef83ad9'),
    returnRawDataArray: abiCoder.encodeParameter('bytes[]', ['0x5ef83ad9']),
    returnGraphData: '0x5ef83ad9',
    expectedResult: '0x5ef83ad9',
  },

  // Case 2
  {
    name: 'TestJSONURL',
    key: '0xd154e1e44d32870ff5ade9e8726fd06d0ed6c996f5946dabfdfd46aa6dd2ea99',
    keyType: 'Singleton',
    valueContent: 'JSONURL',
    valueType: 'bytes',
    // Testing data
    returnRawData: abiCoder.encodeParameter(
      'bytes',

'0x6f357c6a733e78f2fc4a3304c141e8424d02c9069fe08950c6514b27289ead8ef4faa49d6970667 33a2f2f516d6245724b6833466a7341523659 6a73546a485a4e6d364d6344703661527438324674637639414a4a765a6264',
```

```
    ),
    returnRawDataArray: abiCoder.encodeParameter('bytes[]', [

'0x6f357c6a733e78f2fc4a3304c141e8424d02c9069fe08950c6514b27289ead8ef4faa49d6970667333a2f2f51
6d6245724b6833466a73415236596a73546a485a4e6d364d6344703661527438324674637639414a4a765a
6264',
    ]),
    returnGraphData:

'0x6f357c6a733e78f2fc4a3304c141e8424d02c9069fe08950c6514b27289ead8ef4faa49d6970667333a2f2f51
6d6245724b6833466a73415236596a73546a485a4e6d364d6344703661527438324674637639414a4a765a
6264',
    expectedResult: {
      verification: {
        method: 'keccak256(utf8)',
        data: '0x733e78f2fc4a3304c141e8424d02c9069fe08950c6514b27289ead8ef4faa49d', // hash of
stringified json
      },
      url: 'ipfs://QmbErKh3FjsAR6YjsTjHZNm6McDp6aRt82Ftcv9AJJvZbd', // same JSON url from
LSP3Profile below
    },
  },

  // Case 3
  {
    name: 'TestAssetURL',
    key: '0xf18290c9b373d751e12c5ec807278267a807c35c3806255168bc48a85757ceee',
    keyType: 'Singleton',
    valueContent: 'AssetURL',
    valueType: 'bytes',
    // Testing data
    returnRawData: abiCoder.encodeParameter(
      'bytes',

'0x6f357c6aa7d9a84b44013f71356d72e6c15fdc2533c573271c53d053ed8ddcdaa60f4c8169706673 3a2f2f5
16d6245724b6833466a73415236596a73546a485a4e6d364d6344703661527438324674637639414a4a765a
6264',
    ),
    returnRawDataArray: abiCoder.encodeParameter('bytes[]', [

'0x6f357c6aa7d9a84b44013f71356d72e6c15fdc2533c573271c53d053ed8ddcdaa60f4c8169706673 3a2f2f5
16d6245724b6833466a73415236596a73546a485a4e6d364d6344703661527438324674637639414a4a765a
6264',
    ]),
    returnGraphData:

'0x6f357c6aa7d9a84b44013f71356d72e6c15fdc2533c573271c53d053ed8ddcdaa60f4c8169706673 3a2f2f5
16d6245724b6833466a73415236596a73546a485a4e6d364d6344703661527438324674637639414a4a765a
6264',
    expectedResult: {
      verification: {
        method: 'keccak256(utf8)',
```

```
      data: '0xa7d9a84b44013f71356d72e6c15fdc2533c573271c53d053ed8ddcdaa60f4c81', // hash of address '0x0c03fba782b07bcf810deb3b7f0595024a444f4e'
    },
    url: 'ipfs://QmbErKh3FjsAR6YjsTjHZNm6McDp6aRt82Ftcv9AJJvZbd', // FAKE. just used from above TODO: fix this is not an asset URL but a JSON url !!
  },
 },

 // Case 4
 {
  name: 'TestKeccak256',
  key: '0xd6c7198ea09a1d33576888e1dbdf0e07f6cfaf94359e0a4fc11e4f5f1d59d54f4',
  keyType: 'Singleton',
  valueContent: 'Keccak256',
  valueType: 'bytes32',
  // Testing data
  returnRawData: abiCoder.encodeParameter(
    'bytes',
    '0x4d75a97aff0964309140e9821514861e5ddcc827113b70a2b69db16dde0695dc',
  ), // this is bytes
  returnRawDataArray: abiCoder.encodeParameter('bytes[]', [
    '0x4d75a97aff0964309140e9821514861e5ddcc827113b70a2b69db16dde0695dc',
  ]),
  returnGraphData:
    '0x4d75a97aff0964309140e9821514861e5ddcc827113b70a2b69db16dde0695dc',
  expectedResult:
    '0x4d75a97aff0964309140e9821514861e5ddcc827113b70a2b69db16dde0695dc', // 'mytestdata'
 },

 // Case 5
 {
  name: 'TestAddress',
  key: '0x7bf6ecfbf659a88c662d7f099c14e468610f786f6e29f0d346e44f772ef0d187',
  keyType: 'Singleton',
  valueContent: 'Address',
  valueType: 'bytes',
  // Testing data
  returnRawData: abiCoder.encodeParameter(
    'bytes',
    '0x0c03fba782b07bcf810deb3b7f0595024a444f4e',
  ),
  returnRawDataArray: abiCoder.encodeParameter('bytes[]', [
    '0x0c03fba782b07bcf810deb3b7f0595024a444f4e',
  ]),
  returnGraphData: '0x0c03fba782b07bcf810deb3b7f0595024a444f4e',
  expectedResult: '0x0C03fBa782b07bCf810DEb3b7f0595024A444F4e', // a real address
 },

 // Case 6
 {
  name: 'TestMarkdown',
  key: '0x328f991bde3a9d8c548b7b2dbc303a362202dddbcd33219650d85bedcd75ac9b',
```

```
  keyType: 'Singleton',
  valueContent: 'Markdown',
  valueType: 'bytes',
  // Testing data
  returnRawData: abiCoder.encodeParameter(
    'bytes',
    utf8ToHex('# Testing markdown. \n Welcome to markdown **test**.'),
  ),
  returnRawDataArray: abiCoder.encodeParameter('bytes[]', [
    utf8ToHex('# Testing markdown. \n Welcome to markdown **test**.'),
  ]),
  returnGraphData: utf8ToHex(
    '# Testing markdown. \n Welcome to markdown **test**.',
  ),
  expectedResult: '# Testing markdown. \n Welcome to markdown **test**.',
},

// Case 7
{
  name: 'LSP3Name',
  key: '0xa5f15b1fa920bbdbc28f5d785e5224e3a66eb5f7d4092dc9ba82d5e5ae3abc87',
  keyType: 'Singleton',
  valueContent: 'String',
  valueType: 'string',
  // Testing data
  returnRawData: abiCoder.encodeParameter('bytes', utf8ToHex('john-snow')),
  returnRawDataArray: abiCoder.encodeParameter('bytes[]', [
    utf8ToHex('john-snow'),
  ]),
  returnGraphData: utf8ToHex('john-snow'),
  expectedResult: 'john-snow',
},

// Case 8
{
  name: 'LSP3Profile',
  key: '0x5ef83ad95590303e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5',
  keyType: 'Singleton',
  valueContent: 'URL',
  valueType: 'bytes',
  // testing data
  returnRawData: abiCoder.encodeParameter(
    'bytes',
    utf8ToHex('ipfs://QmbErKh3FjsAR6YjsTjHZNm6McDp6aRt82Ftcv9AJJvZbd'),
  ),
  returnRawDataArray: abiCoder.encodeParameter('bytes[]', [
    utf8ToHex('ipfs://QmbErKh3FjsAR6YjsTjHZNm6McDp6aRt82Ftcv9AJJvZbd'),
  ]),
  returnGraphData: utf8ToHex(
    'ipfs://QmbErKh3FjsAR6YjsTjHZNm6McDp6aRt82Ftcv9AJJvZbd',
  ),
  expectedResult: 'ipfs://QmbErKh3FjsAR6YjsTjHZNm6McDp6aRt82Ftcv9AJJvZbd',
```

```
},

// Case 9
{
  name: 'LSP3IssuedAssets[]',
  key: '0x3a47ab5bd3a594c3a8995f8fa58d0876c96819ca4516bd76100c92462f2f9dc0',
  keyType: 'Array',
  valueContent: 'Address',
  valueType: 'address',
  // testing data
  // the full array of values
  returnRawData: [
    abiCoder.encodeParameter('bytes', leftPad(2, 32)), // array length
    abiCoder.encodeParameter(
      'bytes',
      '0xc444009d38d3046bb0cf81fa2cd295ce46a67c78',
    ),
    abiCoder.encodeParameter(
      'bytes',
      '0x4febc3491230571f6e1829e46602e3b110215a2e',
    ),
  ],
  returnRawDataArray: [
    abiCoder.encodeParameter('bytes[]', [leftPad(2, 32)]), // array length
    abiCoder.encodeParameter('bytes[]', [
      '0xc444009d38d3046bb0cf81fa2cd295ce46a67c78',
    ]),
    abiCoder.encodeParameter('bytes[]', [
      '0x4febc3491230571f6e1829e46602e3b110215a2e',
    ]),
  ],
  returnGraphData: [
    leftPad(2, 32), // array length
    '0xc444009d38d3046bb0cf81fa2cd295ce46a67c78',
    '0x4febc3491230571f6e1829e46602e3b110215a2e',
  ],
  expectedResult: [
    '0xc444009d38d3046bb0cF81Fa2Cd295ce46A67C78',
    '0x4fEbC3491230571F6e1829E46602e3b110215A2E',
  ],
},

{
  name: 'LSP3IssuedAssetsWithEmptyValue[]',
  key: '0xbcdf8aea8f803343f50b03205ac25188e17fc1f5e4e42245b0782f68786d9f92',
  keyType: 'Array',
  valueContent: 'Address',
  valueType: 'address',
  // testing data
  // the full array of values
  returnRawData: [
    abiCoder.encodeParameter('bytes', leftPad(2, 32)), // array length
```

```
    abiCoder.encodeParameter('bytes', '0x'),
    abiCoder.encodeParameter(
      'bytes',
      '0x4febc3491230571f6e1829e46602e3b110215a2e',
    ),
  ],
  returnRawDataArray: [
    abiCoder.encodeParameter('bytes[]', [leftPad(2, 32)]),
    abiCoder.encodeParameter('bytes[]', ['0x']),
    abiCoder.encodeParameter('bytes[]', [
      '0x4febc3491230571f6e1829e46602e3b110215a2e',
    ]),
  ],
  returnGraphData: [
    leftPad(2, 32), // array length
    '0x',
    '0x4febc3491230571f6e1829e46602e3b110215a2e',
  ],
  expectedResult: [null, '0x4fEbC3491230571F6e1829E46602e3b110215A2E'],
},

// // Case 10
{
  name: 'TestObjArray[]',
  key: '0x9985edaf12cbacf5ac7d6ed54f0445cc0ea56075aee9b9942e4ab3bf4239f950',
  keyType: 'Array',
  valueContent: 'JSONURL',
  valueType: 'bytes',
  // testing data
  // the full array of values
  returnRawData: [
    abiCoder.encodeParameter('bytes', leftPad(2, 32)), // array length
    abiCoder.encodeParameter(
      'bytes',

'0x6f357c6a733e78f2fc4a3304c141e8424d02c9069fe08950c6514b27289ead8ef4faa49d6970667 33a2f2f51
6d6245724b6833466a7341523659 6a735 46a485a4e6d364d6344703661527438324674637639414a4a765a
6264',
    ),
    abiCoder.encodeParameter(
      'bytes',

'0x6f357c6a81bd0b7ed5ac354abbf24619ce16933f00a4bdfa8fcaf3791d25f69b497abf8869706673 3a2f2f516c
6245724b6833466a7378787878787878787878787878787878787878787878787878787878787639414a4a765a62
64',
    ),
  ],
  returnRawDataArray: [
    abiCoder.encodeParameter('bytes[]', [leftPad(2, 32)]),
    abiCoder.encodeParameter('bytes[]', [

'0x6f357c6a733e78f2fc4a3304c141e8424d02c9069fe08950c6514b27289ead8ef4faa49d697066733a2f2f51
```

```
6d6245724b6833466a73415236596a73546a485a4e6d364d6344703661527438324674637639414a4a765a
6264',
      ]),
      abiCoder.encodeParameter('bytes[]', [

'0x6f357c6a81bd0b7ed5ac354abbf24619ce16933f00a4bdfa8fcaf3791d25f69b497abf8869706673a2f2f516d
6245724b6833466a737878787878787878787878787878787878787878787878787878787639414a4a765a62
64',
      ]),
    ],
    returnGraphData: [
      leftPad(2, 32), // array length

'0x6f357c6a733e78f2fc4a3304c141e8424d02c9069fe08950c6514b27289ead8ef4faa49d697066733a2f2f51
6d6245724b6833466a73415236596a73546a485a4e6d364d6344703661527438324674637639414a4a765a
6264',

'0x6f357c6a81bd0b7ed5ac354abbf24619ce16933f00a4bdfa8fcaf3791d25f69b497abf8869706673a2f2f516d
6245724b6833466a737878787878787878787878787878787878787878787878787878787639414a4a765a62
64',
    ],
    expectedResult: [
      // This JSON from JSONURL above...
      {
        verification: {
          method: 'keccak256(utf8)',
          data: '0x733e78f2fc4a3304c141e8424d02c9069fe08950c6514b27289ead8ef4faa49d', // hash of
stringified json
        },
        url: 'ipfs://QmbErKh3FjsAR6YjsTjHZNm6McDp6aRt82Ftcv9AJJvZbd', // same JSON url from
LSP3Profile below
      },
      // this JSON hash is = {"test1":"value1","test2":"value2","test3":"value3"}
      {
        verification: {
          method: 'keccak256(utf8)',
          data: '0x81bd0b7ed5ac354abbf24619ce16933f00a4bdfa8fcaf3791d25f69b497abf88', // hash of
stringified json
        },
        url: 'ipfs://QmbErKh3Fjsxxxxxxxxxxxxxxxxxxxxxxxxxxxxv9AJJvZbd', // dummy url
      },
    ],
  },

  // // Testing other valueTypes

  // Case 11
  {
    name: 'TestStringValueType',
    key: '0xc0929170bbaeb216f869c80a5c937f7a1c887a5a92262dac50313aef131f0c03',
    keyType: 'Singleton',
    valueContent: 'String',
```

```
  valueType: 'string',
  // Test data
  returnRawData: abiCoder.encodeParameter('bytes', utf8ToHex('Great-string')),
  returnRawDataArray: abiCoder.encodeParameter('bytes[]', [
    utf8ToHex('Great-string'),
  ]),
  returnGraphData: utf8ToHex('Great-string'),
  expectedResult: 'Great-string',
},

// // Case 12
{
  name: 'TestUintValueType',
  key: '0x61529294800f5739edc21a6cf8ba1bad3fd3e11d03d2ab5219ce9c0131b93f93',
  keyType: 'Singleton',
  valueContent: 'Number',
  valueType: 'uint256',
  // Test data
  returnRawData: abiCoder.encodeParameter(
    'bytes',
    '0x0000000000000000000000000000000000000000000000000000000000000063',
  ),
  returnRawDataArray: abiCoder.encodeParameter('bytes[]', [
    '0x0000000000000000000000000000000000000000000000000000000000000063',
  ]),
  returnGraphData:
    '0x0000000000000000000000000000000000000000000000000000000000000063',
  expectedResult: 99,
},

// Case 13
{
  name: 'TestNumberWithBytesValueType',
  key: '0x64a44e72c25d95851b1d449428d8d27093b2ef3e0b36a2b3497ae17edf979e61',
  keyType: 'Singleton',
  valueContent: 'Number',
  valueType: 'bytes',
  // Test data
  returnRawData: abiCoder.encodeParameter(
    'bytes',
    '0x0000000000000000000000000000000000000000000000000000000000000063',
  ),
  returnRawDataArray: abiCoder.encodeParameter('bytes[]', [
    '0x0000000000000000000000000000000000000000000000000000000000000063',
  ]),
  returnGraphData:
    '0x0000000000000000000000000000000000000000000000000000000000000063',
  expectedResult: 99,
},

// Case 14
{
```

```
  name: 'TestStringWithBytesValueType',
  key: '0x3ef4d417afa66557c9e1463723b391a518eee0c61d29be4e10882999c7848041',
  keyType: 'Singleton',
  valueContent: 'String',
  valueType: 'bytes',
  // Test data
  returnRawData: abiCoder.encodeParameter(
    'bytes',
    utf8ToHex('Ok this is a string stored as bytes...'),
  ),
  returnRawDataArray: abiCoder.encodeParameter('bytes[]', [
    utf8ToHex('Ok this is a string stored as bytes...'),
  ]),
  returnGraphData: utf8ToHex('Ok this is a string stored as bytes...'),
  expectedResult: 'Ok this is a string stored as bytes...',
},

// Testing array valueTypes
// Case 15
{
  name: 'TestStringValueTypeArray',
  key: '0xd7a8f1af4a0d9de8d17c177ff06f1689c0c3f1310edbbe537333da0b084ccff18',
  keyType: 'Singleton',
  valueContent: 'String',
  valueType: 'string[]',
  // Testing fields
  returnRawData: abiCoder.encodeParameter(
    'bytes',
    abiCoder.encodeParameter('bytes[]', [
      utf8ToHex('apple sauce'),
      utf8ToHex('butter chicken'),
    ]),
  ),
  returnRawDataArray: abiCoder.encodeParameter('bytes[]', [
    abiCoder.encodeParameter('bytes[]', [
      utf8ToHex('apple sauce'),
      utf8ToHex('butter chicken'),
    ]),
  ]),
  returnGraphData: abiCoder.encodeParameter('bytes[]', [
    utf8ToHex('apple sauce'),
    utf8ToHex('butter chicken'),
  ]),
  expectedResult: ['apple sauce', 'butter chicken'],
},

// Case 16
{
  name: 'TestBytesValueTypeArray',
  key: '0xd6b3622ec62ae4459c0276bd5e2e26011201fada1cbc2b33283e9c20495c05fe',
  keyType: 'Singleton',
  valueContent: 'String',
```

```
  valueType: 'bytes[]',
  // Testing fields
  returnRawData: abiCoder.encodeParameter(
    'bytes',
    abiCoder.encodeParameter('bytes[]', [
      utf8ToHex('apple sauce'),
      utf8ToHex('butter chicken'),
    ]),
  ),
  returnRawDataArray: abiCoder.encodeParameter('bytes[]', [
    abiCoder.encodeParameter('bytes[]', [
      utf8ToHex('apple sauce'),
      utf8ToHex('butter chicken'),
    ]),
  ]),
  returnGraphData: abiCoder.encodeParameter('bytes[]', [
    utf8ToHex('apple sauce'),
    utf8ToHex('butter chicken'),
  ]),
  expectedResult: ['apple sauce', 'butter chicken'],
},

// Case 17
{
  name: 'TestAddressValueTypeArray',
  key: '0xe45f3de809830d5ac3aeab862200fc670391fcb99018dcd2522fee7cf07f93ee',
  keyType: 'Singleton',
  valueContent: 'Address',
  valueType: 'address[]',
  // Testing fields
  returnRawData: abiCoder.encodeParameter(
    'bytes',
    abiCoder.encodeParameter('address[]', [
      '0xCE3e75A43B0A292922199 26EAdC8C5585651219C',
      '0xba61a0b24a228807f23b46064773d28fe51da81c',
    ]),
  ),
  returnRawDataArray: abiCoder.encodeParameter('bytes[]', [
    abiCoder.encodeParameter('address[]', [
      '0xCE3e75A43B0A292922199 26EAdC8C5585651219C',
      '0xba61a0b24a228807f23b46064773d28fe51da81c',
    ]),
  ]),
  returnGraphData: abiCoder.encodeParameter('address[]', [
    '0xCE3e75A43B0A292922199 26EAdC8C5585651219C',
    '0xba61a0b24a228807f23b46064773d28fe51da81c',
  ]),
  expectedResult: [
    '0xCE3e75A43B0A292922199 26EAdC8C5585651219C',
    '0xba61a0b24a228807f23B46064773D28Fe51dA81C',
  ],
},
```

```
// // Case 18
{
  name: 'TestUintValueTypeArray',
  key: '0xdaa41a5e1acc41087359e615 88e80bf0b7f1d96063b98bdff73b4ce3a645b40b',
  keyType: 'Singleton',
  valueContent: 'Number',
  valueType: 'uint256[]',
  // Testing fields
  returnRawData: abiCoder.encodeParameter(
    'bytes',
    abiCoder.encodeParameter('uint256[]', [123, 456]),
  ),
  returnRawDataArray: abiCoder.encodeParameter('bytes[]', [
    abiCoder.encodeParameter('uint256[]', [123, 456]),
  ]),
  returnGraphData: abiCoder.encodeParameter('uint256[]', [123, 456]),
  expectedResult: [
    123, // (firefox metamask key)
    456, // {firefox metamask key}
  ],
},

// Case 19
{
  name: 'TestBytes32ValueTypeArray',
  key: '0x7e2458b2b22ff4357510c3491b7c041df2ee4f11ba4d6f4f4e34101fc2645a97',
  keyType: 'Singleton',
  valueContent: 'Keccak256',
  valueType: 'bytes32[]',
  // Testing fields
  returnRawData: abiCoder.encodeParameter(
    'bytes',
    abiCoder.encodeParameter('bytes32[]', [
      '0xe5d35cae7c9db9879eb8a205baa046ad99503414d6a55eb6725494a4254a6d3f',
      '0x828e919feac2ec05939abd5d221692fbe6bac5667ba5af5d191df1f7ecb1ac21',
    ]),
  ),
  returnRawDataArray: abiCoder.encodeParameter('bytes[]', [
    abiCoder.encodeParameter('bytes32[]', [
      '0xe5d35cae7c9db9879eb8a205baa046ad99503414d6a55eb6725494a4254a6d3f',
      '0x828e919feac2ec05939abd5d221692fbe6bac5667ba5af5d191df1f7ecb1ac21',
    ]),
  ]),
  returnGraphData: abiCoder.encodeParameter('bytes32[]', [
    '0xe5d35cae7c9db9879eb8a205baa046ad99503414d6a55eb6725494a4254a6d3f',
    '0x828e919feac2ec05939abd5d221692fbe6bac5667ba5af5d191df1f7ecb1ac21',
  ]),
  expectedResult: [
    '0xe5d35cae7c9db9879eb8a205baa046ad99503414d6a55eb6725494a4254a6d3f',
    '0x828e919feac2ec05939abd5d221692fbe6bac5667ba5af5d191df1f7ecb1ac21',
  ],
```

```
},

// // Case 20
{
  name: 'TestURLStringValueTypeArray',
  key: '0x1a9818703b62d00000bd3e8c7499296d42966619cd735a92eac7488de8881bb8',
  keyType: 'Singleton',
  valueContent: 'URL',
  valueType: 'string[]',
  // Testing fields
  returnRawData: abiCoder.encodeParameter(
    'bytes',
    abiCoder.encodeParameter('string[]', [
      'ipfs://QmbErKh3Fjsxxxxxxxxxxxxxxxxxxxxxxxxxxv9AJJvZbd',
      'ipfs://QmbErKh3FjsAR6YjsTjHZNm6McDp6aRt82Ftcv9AJJvZbd',
    ]),
  ),
  returnRawDataArray: abiCoder.encodeParameter('bytes[]', [
    abiCoder.encodeParameter('string[]', [
      'ipfs://QmbErKh3Fjsxxxxxxxxxxxxxxxxxxxxxxxxxxv9AJJvZbd',
      'ipfs://QmbErKh3FjsAR6YjsTjHZNm6McDp6aRt82Ftcv9AJJvZbd',
    ]),
  ]),
  returnGraphData: abiCoder.encodeParameter('string[]', [
    'ipfs://QmbErKh3Fjsxxxxxxxxxxxxxxxxxxxxxxxxxxv9AJJvZbd',
    'ipfs://QmbErKh3FjsAR6YjsTjHZNm6McDp6aRt82Ftcv9AJJvZbd',
  ]),
  expectedResult: [
    'ipfs://QmbErKh3Fjsxxxxxxxxxxxxxxxxxxxxxxxxxxv9AJJvZbd',
    'ipfs://QmbErKh3FjsAR6YjsTjHZNm6McDp6aRt82Ftcv9AJJvZbd',
  ],
},

// Case 21
{
  name: 'TestHashKey',
  key: '0xed579debad05d91a79b465899871871dfce1c8ffa8b1d8c1ddc851cc104ea6029',
  keyType: 'Singleton',
  valueContent:
    '0x9c22ff5f21f0b81b113e63f7db6da94fedef11b2119b4088b89664fb9a3cb658', // keccak hash of 'test'
  valueType: 'bytes32',
  // Testing data
  returnRawData: abiCoder.encodeParameter(
    'bytes',
    '0x9c22ff5f21f0b81b113e63f7db6da94fedef11b2119b4088b89664fb9a3cb658',
  ), // this is bytes
  returnRawDataArray: abiCoder.encodeParameter('bytes[]', [
    '0x9c22ff5f21f0b81b113e63f7db6da94fedef11b2119b4088b89664fb9a3cb658',
  ]),
  returnGraphData:
    '0x9c22ff5f21f0b81b113e63f7db6da94fedef11b2119b4088b89664fb9a3cb658',
  expectedResult:
```

```
    '0x9c22ff5f21f0b81b113e63f7db6da94fedef11b2119b4088b89664fb9a3cb658', // 'mytestdata'
  },

  {
    name: 'MyCoolAddress:0xcafecafecafecafecafecafecafecafecafecafe',
    key: '0x22496f48a493035f0ab40000cafecafecafecafecafecafecafecafecafecafe',
    keyType: 'Mapping',
    valueContent: '0x5ef83ad9',
    valueType: 'bytes',
    // Testing data
    returnRawData: abiCoder.encodeParameter('bytes', '0x5ef83ad9'),
    returnRawDataArray: abiCoder.encodeParameter('bytes[]', ['0x5ef83ad9']),
    returnGraphData: '0x5ef83ad9',
    expectedResult: '0x5ef83ad9',
  },
  {
    name: 'AddressPermissions:Permissions:cafecafecafecafecafecafecafecafecafecafe',
    key: '0x4b80742de2bf82acb3630000cafecafecafecafecafecafecafecafecafecafe',
    keyType: 'MappingWithGrouping',
    valueContent: '0x5ef83ad9',
    valueType: 'bytes',
    // Testing data
    returnRawData: abiCoder.encodeParameter('bytes', '0x5ef83ad9'),
    returnRawDataArray: abiCoder.encodeParameter('bytes[]', ['0x5ef83ad9']),
    returnGraphData: '0x5ef83ad9',
    expectedResult: '0x5ef83ad9',
  },
  {
    name: 'Hello:<address>',
    key: '0x06b3dfaec148fb1bb2b00000cafecafecafecafecafecafecafecafecafecafe', // encoded for cafecafe...
address - parameters are bellow
    dynamicKeyParts: ['0xcafecafecafecafecafecafecafecafecafecafe'],
    keyType: 'Singleton',
    valueContent: 'AssetURL',
    valueType: 'bytes',
    // Testing data
    returnRawData: abiCoder.encodeParameter(
      'bytes',

'0x6f357c6aa7d9a84b44013f71356d72e6c15fdc2533c573271c53d053ed8ddcdaa60f4c81697066733a2f2f5
16d6245724b6833466a73415236596a73546a485a4e6d364d6344703661527438324674637639414a4a765a
6264',
    ),
    returnRawDataArray: abiCoder.encodeParameter('bytes[]', [

'0x6f357c6aa7d9a84b44013f71356d72e6c15fdc2533c573271c53d053ed8ddcdaa60f4c81697066733a2f2f5
16d6245724b6833466a73415236596a73546a485a4e6d364d6344703661527438324674637639414a4a765a
6264',
    ]),
    returnGraphData: '0x0c03fba782b07bcf810deb3b7f0595024a444f4e',
    expectedResult: {
      verification: {
```

```
      method: 'keccak256(utf8)',
      data: '0xa7d9a84b44013f71356d72e6c15fdc2533c573271c53d053ed8ddcdaa60f4c81', // hash of address '0x0c03fba782b07bcf810deb3b7f0595024a444f4e'
    },
    url: 'ipfs://QmbErKh3FjsAR6YjsTjHZNm6McDp6aRt82Ftcv9AJJvZbd', // FAKE. just used from above TODO: fix this is not an asset URL but a JSON url !!
  },
 },
 {
  name: 'TestStringWithBytes4ValueContent',
  key: '0xb61b0a1d86687ef022781d2698d5e0221997458e3a720cded0b8f165a029d3c5',
  keyType: 'Singleton',
  valueContent: 'Bytes4',
  valueType: 'bytes',
  // Test data
  returnRawData: abiCoder.encodeParameter('bytes', '0xcafecafe'),
  returnRawDataArray: abiCoder.encodeParameter('bytes[]', ['0xcafecafe']),
  expectedResult: '0xcafecafe',
  returnGraphData: '0xcafecafe',
 },
 {
  name: 'TestStringWithBytes32ValueType',
  key: '0xbaced8d1d0b02d5f412674cac7ad60f0f3e8ae29f2b8d4ad463fa1f5fc103d4d',
  keyType: 'Singleton',
  valueContent: 'Bytes32',
  valueType: 'bytes32',
  // Test data
  returnRawData: abiCoder.encodeParameter(
   'bytes',
   '0xcafecafecafecafecafecafecafecafecafecafecafecafecafecafecafecafe',
  ),
  returnRawDataArray: abiCoder.encodeParameter('bytes[]', [
   '0xcafecafecafecafecafecafecafecafecafecafecafecafecafecafecafecafe',
  ]),
  expectedResult:
   '0xcafecafecafecafecafecafecafecafecafecafecafecafecafecafecafecafe',
  returnGraphData:
   '0xcafecafecafecafecafecafecafecafecafecafecafecafecafecafecafecafe',
 },
 {
  name: 'TestStringWithBytes4ValueType',
  key: '0x1b92e269c7ce7fc16e625562aa588403fe603edb4e2740b0558ed44faa3c1728',
  keyType: 'Singleton',
  valueContent: 'Bytes4',
  valueType: 'bytes4',
  // Test data
  returnRawData: abiCoder.encodeParameter('bytes', '0xcafecafe'),
  returnRawDataArray: abiCoder.encodeParameter('bytes[]', ['0xcafecafe']),
  expectedResult: '0xcafecafe',
  returnGraphData: '0xcafecafe',
 },
```

```
// Nested array tests
// NOTE: The below are failing on decode all (no source loops, or individual) with data out-of-bounds
// Case 22
// {
//   "name": "TestNestedArrayStringArray[]",
//   "key": "0xe4d20147947de3d1b329a854199dbf938a8f8c375c411fa86df1820d4cca043b",
//   "keyType": "Array",
//   "valueContent": "String",
//   "valueType": "string[]",
//   // testing data
//   // the full array of values
//   "returnRawData": [
//
"0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000002000000000000000000000000000000000000000000000000000000000000000300000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000016000000000000000000000000000000000000000000000000000000000000002000000000000000000000000000000000000000000000000000000000000003000000000000000000000000000000000000000000000000000000000000006000000000000000000000000000000000000000000000000000000000000000a000000000000000000000000000000000000000000000000000000000000000e000000000000000000000000000000000000000000000000000000000000000a737472696e672d6f6e650000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000a737472696e672d74776f0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000c737472696e672d7468726565000000000000000000000000000000000000000000",
//
"0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000001000000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000004000000000000000000000000000000000000000000000000000000000000000800000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000b737472696e672d666f7572000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000b737472696e672d6669766500000000000000000000000000000000",
//
"0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000000a000000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000010000000000000000000000000000000000000000000000000000000000000002000000000000000000000000000000000000000000000000000000000000000a737472696e672d7369780000000000000000000000000000000000000000",
//   ],
//   "returnGraphData": [
//     "0x0000000000000000000000000000000000000000000000000000000000000003", // array length
//
"0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000003000000000000000000000000000000000000000000000000000000000000000600000000000000000000000000000000000000000000000000000000000000a000000000000000000000000000000000000000000000000000000000000000e000000000000000000000000000000000000000000000000000000000000000a737472696e672d6f6e650000000000000000000000000000
```

```
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000a7
37472696e672d74776f0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000c737472696e672d7468726565000000000000000000000000
0000000000000000",
  //
"0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000
0000000000000000000000000000000000000000000000200000000000000000000000000000000000000000
0000000000000000000040000000000000000000000000000000000000000000000000000000000000000008
0000000000000000000000000000000000000000000000000000000000000000b737472696e672d666f757
2000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000b737472696e672d6669766500000000000000000000000000000000000000000000",
  //
"0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000
0000000000000000000000000000000000000000000000100000000000000000000000000000000000000000
0000000000000000000020000000000000000000000000000000000000000000000000000000000000000000
a737472696e672d736978000000000000000000000000000000000000000000",
  //   ],
  //   "expectedResult": [
  //     [ 'string-one', 'string-two', 'string-three' ],
  //     [ 'string-four', 'string-five', ],
  //     [ 'string-six' ]
  //   ]

  // },

  // Case 23
  // {
  //   "name": "TestNestedArrayObjArray[]",
  //   "key": "0xf9a149b854fc24a5b7c91cce364e1d8306c7d78438a7fdc5c729c394320ebcbe",
  //   "keyType": "Array",
  //   "valueContent": "JSONURL",
  //   "valueType": "bytes[]",
  //   // testing data
  //   // the full array of values
  //   "returnRawData": [
  //
"0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000
0000000000000000000000000000000000000000000000200000000000000000000000000000000000000000
0000000000000000000003", // array length
  //
"0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000
0000000000000000000000000000000000000000000001c0000000000000000000000000000000000000000000
0000000000000000000020000000000000000000000000000000000000000000000000000000000000000000
3000000000000000000000000000000000000000000000000000000000000000600000000000000000000000
0000000000000000000000000000000000000000c0000000000000000000000000000000000000000000000000
0000000000000000012000000000000000000000000000000000000000000000000000000000000000000396
f357c6acadb1115f687654d9b8077b1c0b8553f294e54862cf17e6afc0cade73dc35c88697066733a2f2f51773
039656a66646b6a667878780000000000000000000000000000000000000000000000000000000000000000
0000000000000396f357c6ab0bf98bf562a0191622eb530e185cff88dc086b92faa27cb535513a62591a5f7697
066733a2f2f51773039656a66646b6a667979790000000000000000000000000000000000000000000000000
0000000000000000000000000000396f357c6ad74263f10a321b733c6dc6aea656e383ce31d653814d1bcb3
5630729d2bf993c697066733a2f2f51773039656a66646b6a667a7a7a00000000000000",
```

```
  //
"0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000
0000000000000000000000000000000000000000000000140000000000000000000000000000000000000000000000
0000000000000000000000020000000000000000000000000000000000000000000000000000000000000000000000
2000000000000000000000000000000000000000000000000000000000000000400000000000000000000000
0000000000000000000000000000000000000000a000000000000000000000000000000000000000000000
0000000000000000000396f357c6acadb1115f687654d9b8077b1c0b8553f294e54862cf17e6afc0cade73dc3
5c88697066733a2f2f51773039656a66646b6a667878780000000000000000000000000000000000000000
0000000000000000000000000000000000000396f357c6ab0bf98bf562a0191622eb530e185cff88dc086b92faa
27cb535513a62591a5f7697066733a2f2f51773039656a66646b6a6679797900000000000000",
  //
"0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000
0000000000000000000000000000000000000000c000000000000000000000000000000000000000000000
0000000000000000000020000000000000000000000000000000000000000000000000000000000000000000000
1000000000000000000000000000000000000000000000000000000000000020000000000000000000000
0000000000000000000000000000000000000000396f357c6acadb1115f687654d9b8077b1c0b8553f294e5
4862cf17e6afc0cade73dc35c88697066733a2f2f51773039656a66646b6a667878780000000000000000",
  //   ],
  //   "returnGraphData": [
  //     "0x0000000000000000000000000000000000000000000000000000000000000003", // array length
  //
"0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000
0000000000000000000000000000000000000000300000000000000000000000000000000000000000000000
0000000000000000000060000000000000000000000000000000000000000000000000000000000000000000000c
0000000000000000000000000000000000000000000000000000000000000012000000000000000000000
0000000000000000000000000000000000000000396f357c6acadb1115f687654d9b8077b1c0b8553f294e5
4862cf17e6afc0cade73dc35c88697066733a2f2f51773039656a66646b6a667878780000000000000000000
0000000000000000000000000000000000000000000000000000396f357c6ab0bf98bf562a0191622e
b530e185cff88dc086b92faa27cb535513a62591a5f7697066733a2f2f51773039656a66646b6a66797979000
0000000000000000000000000000000000000000000000000000000000000000000000396f357c6ad74
263f10a321b733c6dc6aea656e383ce31d653814d1bcb35630729d2bf993c697066733a2f2f51773039656a6
6646b6a667a7a7a00000000000000",
  //
"0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000
0000000000000000000000000000000000000000200000000000000000000000000000000000000000000000
0000000000000000000040000000000000000000000000000000000000000000000000000000000000000000000a
0000000000000000000000000000000000000000000000000000000000000000396f357c6acadb1115f68765
4d9b8077b1c0b8553f294e54862cf17e6afc0cade73dc35c88697066733a2f2f51773039656a66646b6a66787
8780000000000000000000000000000000000000000000000000000000000000000000396f357
c6ab0bf98bf562a0191622eb530e185cff88dc086b92faa27cb535513a62591a5f7697066733a2f2f517730396
56a66646b6a667979790000000000000",
  //
"0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000
0000000000000000000000000000000000000000100000000000000000000000000000000000000000000000
0000000000000000000020000000000000000000000000000000000000000000000000000000000000000000003
96f357c6acadb1115f687654d9b8077b1c0b8553f294e54862cf17e6afc0cade73dc35c88697066733a2f2f517
73039656a66646b6a667878780000000000000",
  //   ],
  //   "expectedResult": [
  //     // This JSON from JSONURL above...
  //     [
```

```
  //      {
  //        "hashFunction": "keccak256(utf8)",
  //        "jsonHash": "0xcadb1115f687654d9b8077b1c0b8553f294e54862cf17e6afc0cade73dc35c88", //
hash of stringified json = {test1:'value1',test2:'value2'}
  //        "jsonURL": "ipfs://Qw09ejfdkjfxxx" // dummy url
  //      },

  //      {
  //        "hashFunction": "keccak256(utf8)",
  //        "jsonHash": "0xb0bf98bf562a0191622eb530e185cff88dc086b92faa27cb535513a62591a5f7", //
hash of stringified json = {test3:'value3',test4:'value4'}
  //        "jsonURL": "ipfs://Qw09ejfdkjfyyy" // dummy url
  //      },
  //      {
  //        "hashFunction": "keccak256(utf8)",
  //        "jsonHash": "0xd74263f10a321b733c6dc6aea656e383ce31d653814d1bcb35630729d2bf993c", //
hash of stringified json = {test5:;value5',test6:'value6'}
  //        "jsonURL": "ipfs://Qw09ejfdkjfzzz" // dummy url
  //      }
  //      // encoded array:
0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000
0000000000000000000000000000000000000000000000030000000000000000000000000000000000000000000000000000000000
0000000000000000000600000000000000000000000000000000000000000000000000000000000000000000000000c0
0000000000000000000000000000000000000000000000000000000000000120000000000000000000000000
0000000000000000000000000000000000000396f357c6acadb1115f687654d9b8077b1c0b8553f294e54
862cf17e6afc0cade73dc35c88697066733a2f2f51773039656a66646b6a66787878000000000000000000000000
0000000000000000000000000000000000000000000000000000000396f357c6ab0bf98bf562a0191622eb
530e185cff88dc086b92faa27cb535513a62591a5f7697066733a2f2f51773039656a66646b6a667979790000
0000000000000000000000000000000000000000000000000000000000000000000000000396f357c6ad742
63f10a321b733c6dc6aea656e383ce31d653814d1bcb35630729d2bf993c697066733a2f2f51773039656a66
646b6a667a7a7a00000000000000

  //    ],
  //    [

  //      {
  //        "hashFunction": "keccak256(utf8)",
  //        "jsonHash": "0xcadb1115f687654d9b8077b1c0b8553f294e54862cf17e6afc0cade73dc35c88", //
hash of stringified json = {test1:'value1',test2:'value2'}
  //        "jsonURL": "ipfs://Qw09ejfdkjfxxx" // dummy url
  //      },

  //      {
  //        "hashFunction": "keccak256(utf8)",
  //        "jsonHash": "0xb0bf98bf562a0191622eb530e185cff88dc086b92faa27cb535513a62591a5f7", //
hash of stringified json = {test3:'value3',test4:'value4'}
  //        "jsonURL": "ipfs://Qw09ejfdkjfyyy" // dummy url
  //      },
  //      // encoded:
0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000
0000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000
0000000000000000000400000000000000000000000000000000000000000000000000000000000000000000000000a0
```

```
0000000000000000000000000000000000000000000000000000000000000396f357c6acadb1115f687654
d9b8077b1c0b8553f294e54862cf17e6afc0cade73dc35c88697066733a2f2f51773039656a66646b6a667878
7800000000000000000000000000000000000000000000000000000000000000000000000000000000000396f357c
6ab0bf98bf562a0191622eb530e185cff88dc086b92faa27cb535513a62591a5f7697066733a2f2f5177303965
6a66646b6a6679797900000000000000

  //     ],
  //     [

  //       {
  //         "hashFunction": "keccak256(utf8)",
  //         "jsonHash": "0xcadb1115f687654d9b8077b1c0b8553f294e54862cf17e6afc0cade73dc35c88", //
hash of stringified json = {test1:'value1',test2:'value2'}
  //         "jsonURL": "ipfs://Qw09ejfdkjfxxx" // dummy url
  //       },
  //       // encoded:
0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000
0000000000000000000000000000000000000000000000010000000000000000000000000000000000000000000
0000000000000000000020000000000000000000000000000000000000000000000000000000000000039
6f357c6acadb1115f687654d9b8077b1c0b8553f294e54862cf17e6afc0cade73dc35c88697066733a2f2f5177
3039656a66646b6a6678787800000000000000

  //     ]
  //   ]

  // },
  // Case xx
  // NOTE: This test can not be made to be symmetrical
  // {
  //   "name": "TestNotSetArray[]",
  //   "key": "0xa90924b9b409ab609ed5136d82249ce17851a12b49bbb6f20fdedd6b3a2be61b",
  //   "keyType": "Array",
  //   "valueContent": "Address",
  //   "valueType": "address",
  //   // testing data
  //   // the full array of values
  //   "returnRawData": [
  //     "0x", // array length
  //   ],
  //   "returnGraphData": [
  //     "0x", // array length
  //   ],
  //   "expectedResult" :[
  //   ]

  // },
];
```

</file>

<file>

path: /test/testHelpers.ts

url: https://github.com/ERC725Alliance/erc725.js/blob/main/test/testHelpers.ts

```
/*
    This file is part of @erc725/erc725.js.
    @erc725/erc725.js is free software: you can redistribute it and/or modify
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
    @erc725/erc725.js is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.
    You should have received a copy of the GNU Lesser General Public License
    along with @erc725/erc725.js.  If not, see <http://www.gnu.org/licenses/>.
*/
/**
 * @file test/testHelpers.ts
 * @author Robert McLeod <@robertdavid010>, Fabian Vogelsteller <fabian@lukso.network>
 * @date 2020
 */

import { encodeArrayKey } from '../src/lib/utils';

/**
 * Takes the schema object and builds a full dataset as per expected from provider.
 */
export function generateAllRawData(schema, isArrayMode: boolean) {
  const results: { key: string; value: string }[] = [];
  for (let index = 0; index < schema.length; index++) {
    const element = schema[index];
    // if is array push data
    if (element.keyType === 'Array') {
      const correctReturnRawData = isArrayMode
        ? element.returnRawDataArray
        : element.returnRawData;

      correctReturnRawData.forEach((e, i) => {
        // we assume always first element in the array in returnData array is the length
        if (i === 0) {
          results.push({
            key: element.key,
            value: e,
          });
        } else {
          // This is array length key/value pair
          results.push({
            key: encodeArrayKey(element.key, i - 1),
            value: e,
          });
```

```
      }
    });
  } else {
    results.push({
      key: element.key,
      value: isArrayMode ? element.returnRawDataArray : element.returnRawData,
    });
  }
 }

 return results;
}

/**
 * Takes the schema object and builds a full dataset as per expected from provider.
 */
export function generateAllData(schema) {
 const results: any[] = [];
 for (let index = 0; index < schema.length; index++) {
  const element = schema[index];

  // if is a 'nested' array, need to flatten it, and add {key,value} elements
  if (element.keyType === 'Array') {
    element.returnGraphData.forEach((e, i) => {
      if (i === 0) {
        // We need the new key, and to 'flatten the array as per expected from chain data
        if (e) {
          results.push({
            key: element.key,
            value: e.toLowerCase(), // force address to lowercase
          }); // we subtract one from length because this has the extra array length key in the array
        }
      } else {
        // This is array length key/value pair
        results.push({
          key: encodeArrayKey(element.key, i - 1),
          value: e ? e.toLowerCase() : e, // force address to lowercase
        });
      }
    }); // end .forEach()
  } else {
    results.push({
      key: element.key,
      value: element.returnGraphData,
    });
  }
 }

 return results;
}

/**
```

```
 * Takes the test schema/cases and builds full expected results.
 * Removes dynamic keys.
 */
export function generateAllResults(schemas) {
  return schemas
    .filter((schema) => !schema.dynamicKeyParts)
    .map((schema) => {
      return {
        name: schema.name,
        key: schema.key,
        value: schema.expectedResult,
      };
    });
}
```

</file>

<file>

path: /tsconfig.json

url: https://github.com/ERC725Alliance/erc725.js/blob/main/tsconfig.json

```
{
  "compilerOptions": {
    "incremental": true,
    "target": "es2017",
    "outDir": "build/main",
    "rootDirs": ["src", "schemas"],
    "moduleResolution": "node",
    "module": "commonjs",
    "declaration": true,
    "strict": true,
    "noImplicitAny": false, //TODO: this should be removed
    "esModuleInterop": true,
    "inlineSourceMap": false,
    "sourceMap": true,
    "resolveJsonModule": true /* Include modules imported with .json extension. */,

    /* Additional Checks */
    "noUnusedLocals": true /* Report errors on unused locals. */,
    "noUnusedParameters": true /* Report errors on unused parameters. */,
    "noImplicitReturns": true /* Report error when not all code paths in function return a value. */,
    "noFallthroughCasesInSwitch": true /* Report errors for fallthrough cases in switch statement. */,

    /* Debugging Options */
    "traceResolution": false /* Report module resolution log messages. */,
    "listEmittedFiles": false /* Print names of generated files part of the compilation. */,
    "listFiles": false /* Print names of files part of the compilation. */,
    "pretty": true /* Stylize errors and messages using color and context. */,

    "lib": ["es2019", "dom"],
    "types": ["mocha", "node"],
    "typeRoots": ["node_modules/@types", "src/types"]
  },
  "include": ["src", "schemas"],
  "exclude": ["node_modules", "tests/*", "**/*.test.ts"],
  "compileOnSave": false
}
```

</file>

<file>

path: /tsconfig.module.json

url: https://github.com/ERC725Alliance/erc725.js/blob/main/tsconfig.module.json

```
{
  "extends": "./tsconfig",
  "compilerOptions": {
    "target": "es2015",
    "outDir": "build/module",
    "module": "es2015"
  }
}
```

</file>

<file>

path: /typedoc.json

url: https://github.com/ERC725Alliance/erc725.js/blob/main/typedoc.json

```
{
  "entryPoints": ["./src/index.ts"],
  "out": "docs/html",
  "includeVersion": true,
  "plugin": [],
  "excludePrivate": true,
  "readme": "none",
  "exclude": ["**/*+(.specl.e2e).ts"]
}
```

</file>

</repository>

<repository>

salted-ups-contracts

overview

repository: https://github.com/b00ste/salted-ups-contracts/
userName: b00ste
repository: salted-ups-contracts
branch: main
date: 2023-11-27T10:50:57+01:00 (1701078657)

<file>

path: /.prettierrc

url: https://github.com/b00ste/salted-ups-contracts/blob/main/.prettierrc

```
{
    "tabWidth": 4,
    "useTabs": true,
    "singleQuote": true,
    "trailingComma": "all"
}
```

</file>

<file>

path: /README.md

url: https://github.com/b00ste/salted-ups-contracts/blob/main/README.md

Sample Hardhat Project

This project demonstrates a basic Hardhat use case. It comes with a sample contract, a test for that contract, and a script that deploys that contract.

Try running some of the following tasks:

```
npx hardhat help
npx hardhat test
REPORT_GAS=true npx hardhat test
npx hardhat node
npx hardhat run scripts/deploy.ts
```

salted-ups-contracts

</file>

<file>

path: /contracts/LSP16.sol

url: https://github.com/b00ste/salted-ups-contracts/blob/main/contracts/LSP16.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.17;

import {LSP16UniversalFactory} from "@lukso/lsp-smart-contracts/contracts/LSP16UniversalFactory/LSP16UniversalFactory.sol";
```

</file>

<file>

path: /contracts/SaltedUniversalProfileFactory.sol

url: https://github.com/b00ste/salted-ups-contracts/blob/main/contracts/SaltedUniversalProfileFactory.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.22;

// utils
import {LSP2Utils} from "@lukso/lsp-smart-contracts/contracts/LSP2ERC725YJSONSchema/LSP2Utils.sol";

// constants
import  {
    _LSP1_UNIVERSAL_RECEIVER_DELEGATE_KEY
} from "@lukso/lsp-smart-contracts/contracts/LSP1UniversalReceiver/LSP1Constants.sol";
import {
    _LSP3_SUPPORTED_STANDARDS_KEY,
    _LSP3_SUPPORTED_STANDARDS_VALUE,
    _LSP3_PROFILE_KEY
} from "@lukso/lsp-smart-contracts/contracts/LSP3ProfileMetadata/LSP3Constants.sol";
import {
    _LSP6KEY_ADDRESSPERMISSIONS_ARRAY,
    _LSP6KEY_ADDRESSPERMISSIONS_ARRAY_PREFIX,
    _LSP6KEY_ADDRESSPERMISSIONS_PERMISSIONS_PREFIX,
    ALL_REGULAR_PERMISSIONS
} from "@lukso/lsp-smart-contracts/contracts/LSP6KeyManager/LSP6Constants.sol";

// modules
import {ILSP23LinkedContractsFactory} from "@lukso/lsp-smart-contracts/contracts/LSP23LinkedContractsFactory/ILSP23LinkedContractsFactory.sol";
import {LSP23LinkedContractsFactory} from "@lukso/lsp-smart-contracts/contracts/LSP23LinkedContractsFactory/LSP23LinkedContractsFactory.sol";
import {UniversalProfileInit} from "@lukso/lsp-smart-contracts/contracts/UniversalProfileInit.sol";
import {UniversalProfile} from "@lukso/lsp-smart-contracts/contracts/UniversalProfile.sol";
import {LSP6KeyManagerInit} from "@lukso/lsp-smart-
```

```
contracts/contracts/LSP6KeyManager/LSP6KeyManagerInit.sol";

// errors
error CallerNotUniversalProfileOwner(address caller, address universalProfile);

// events
event SaltedUniversalProfileDeployed(
    address indexed deployer,
    address indexed universalProfileAddress
);
event SaltedUniversalProfileExported(
    address indexed deployer,
    address indexed universalProfileAddress
);

contract SaltedUniversalProfileFactory {
    /// ------ Constants - DO NOT CHANGE ------
    address constant LSP23_ADDRESS = 0x2300000A84D25dF63081feAa37ba6b62C4c89a30;
    address constant POST_DEPLOYMENT_MODULE_ADDRESS =
0x000000000066093407b6704B89793beFfD0D8F00;
    address constant UP_INIT_ADDRESS = 0x52c90985AF970D4E0DC26Cb5D052505278aF32A9;
    address constant KM_INIT_ADDRESS = 0xa75684d7D048704a2DB851D05Ba0c3cbe226264C;
    address constant URD_UP_ADDRESS  = 0xA5467dfe7019bF2C7C5F7A707711B9d4cAD118c8;
    bytes constant URD_PERMISSIONS =
hex"0000000000000000000000000000000000000000000000000000000000060080";
    /// ---------------------------------------

    /**
     * @dev Used to store the owner of each deployed up.
     */
    mapping (address => address) private universalProfilesOwners;
    mapping (address => address[]) private deployedUniversalProfiles;
    mapping (address => address[]) private exportedUniversalProfiles;

    /**
     * @dev Gatekeeper, doen't allow modifying the Universal Profile unless the caller is the one that
deployed it.
     */
    modifier OnlyUniversalProfileOwner(address universalProfileAddress) {
        if (msg.sender != universalProfilesOwners[universalProfileAddress]) {
            revert CallerNotUniversalProfileOwner(msg.sender, universalProfileAddress);
        }
        _;
    }

    /**
     * @notice Deployed a       (salted) Universal Profile.
     *
     * @dev Deploy a Universal Profile owned by a Key Manager and make this contract the main controller.
     *
     * @custom:info
     * - The deployed Universal Profile will also have a default URD set with some permissions.
```

```
    * - The deployed Unviersal Profile also supports LSP3 Profile Metadata, but does not have the metadata set at this point.
    *
    * @param salt A 32 bytes value used to deploy the Universal Profile at a pre-deterministic address.
    *
    * @return universalProfileAddress The address of the       (salted) Universal Profile.
    */
   function deploy(bytes32 salt) public payable returns(address universalProfileAddress) {
       address caller = msg.sender;

       /// ------ Data for Universal Profile deployment ------
       ILSP23LinkedContractsFactory.PrimaryContractDeploymentInit memory primaryContractDeploymentInit = ILSP23LinkedContractsFactory.PrimaryContractDeploymentInit({
           salt: salt,
           fundingAmount: 0,
           implementationContract: UP_INIT_ADDRESS,
           initializationCalldata: abi.encodeCall(UniversalProfileInit.initialize, POST_DEPLOYMENT_MODULE_ADDRESS)
       });
       /// ---------------------------------------------------

       /// ------ Data for Key Manager deployment ------
       ILSP23LinkedContractsFactory.SecondaryContractDeploymentInit memory secondaryContractDeploymentInit = ILSP23LinkedContractsFactory.SecondaryContractDeploymentInit({
           fundingAmount: 0,
           implementationContract: KM_INIT_ADDRESS,
           addPrimaryContractAddress: true,
           initializationCalldata: abi.encodePacked(LSP6KeyManagerInit.initialize.selector),
           extraInitializationParams: ""
       });
       /// ---------------------------------------------

       /// ------ Encode Data Keys & Values for updating permissions & LSP3Metadata ------
       bytes32[] memory dataKeys = new bytes32[](7);
       // ------ LSP1 ------
       dataKeys[0] = _LSP1_UNIVERSAL_RECEIVER_DELEGATE_KEY;
       // ------------------

       // ------ LSP3 ------
       dataKeys[1] = _LSP3_SUPPORTED_STANDARDS_KEY;
       // ------------------

       // ------ LSP6 ------
       dataKeys[2] = _LSP6KEY_ADDRESSPERMISSIONS_ARRAY;
       dataKeys[3] = LSP2Utils.generateArrayElementKeyAtIndex(
           _LSP6KEY_ADDRESSPERMISSIONS_ARRAY,
           0
       );
       dataKeys[4] = LSP2Utils.generateMappingKey(
           _LSP6KEY_ADDRESSPERMISSIONS_PERMISSIONS_PREFIX,
           bytes20(address(this))
       );
```

```
        dataKeys[5] = LSP2Utils.generateArrayElementKeyAtIndex(
            _LSP6KEY_ADDRESSPERMISSIONS_ARRAY,
            1
        );
        dataKeys[6] = LSP2Utils.generateMappingKey(
            _LSP6KEY_ADDRESSPERMISSIONS_PERMISSIONS_PREFIX,
            bytes20(URD_UP_ADDRESS)
        );
        // ------------------

        bytes[] memory dataValues = new bytes[](7);
        // ------ LSP1 ------
        dataValues[0] = abi.encodePacked(URD_UP_ADDRESS);
        // ------------------

        // ------ LSP3 ------
        dataValues[1] = abi.encodePacked(_LSP3_SUPPORTED_STANDARDS_VALUE);
        // ------------------

        // ------ LSP6 ------
        dataValues[2] = abi.encodePacked(bytes16(uint128(2)));
        dataValues[3] = abi.encodePacked(address(this));
        dataValues[4] = abi.encodePacked(ALL_REGULAR_PERMISSIONS);
        dataValues[5] = abi.encodePacked(URD_UP_ADDRESS);
        dataValues[6] = URD_PERMISSIONS;
        // ------------------

        bytes memory postDeploymentModuleCalldata = abi.encode(
            dataKeys,
            dataValues
        );
        /// ---------------------------------------------------------------------------

        /// ------ Deploy the Universal Profile with Key Manager ------
        (universalProfileAddress, ) = LSP23LinkedContractsFactory(LSP23_ADDRESS)
            .deployERC1167Proxies(
                primaryContractDeploymentInit,
                secondaryContractDeploymentInit,
                POST_DEPLOYMENT_MODULE_ADDRESS,
                postDeploymentModuleCalldata
            );
        /// -----------------------------------------------------------

        universalProfilesOwners[universalProfileAddress] = caller;
        deployedUniversalProfiles[caller].push(universalProfileAddress);

        emit SaltedUniversalProfileDeployed(caller, universalProfileAddress);
    }

    /**
     * @dev Add the LSP3 Profile Metadata for the Unviersal Profile and change the main controller from this
address to a new address.
```

```
    *
    * @custom:warning The caller must be the deployer of the Universal Profile at address:
&amp;nbsp;universalProfileAddress&amp;nbsp;.
    *
    * @param universalProfileAddress The address of the Universal profile that was deployed using this
contract.
    * @param newMainController The controller that will replace this contract for the Universal Profile.
    * @param LSP3ProfileMetadata Universal Profile metadata.
    */
   function changeMainController(
      address universalProfileAddress,
      address newMainController,
      bytes memory LSP3ProfileMetadata
   ) public OnlyUniversalProfileOwner(universalProfileAddress) {
      /// ------ Encode Data Keys & Values for updating permissions & LSP3Metadata ------
      bytes32[] memory dataKeys = new bytes32[](4);
      // ------ LSP3 ------
      dataKeys[0] = _LSP3_PROFILE_KEY;
      // ------------------

      // ------ LSP6 ------
      // Adding permissions for the new main controller
      dataKeys[1] = LSP2Utils.generateArrayElementKeyAtIndex(
         _LSP6KEY_ADDRESSPERMISSIONS_ARRAY,
         0
      );
      dataKeys[2] = LSP2Utils.generateMappingKey(
         _LSP6KEY_ADDRESSPERMISSIONS_PERMISSIONS_PREFIX,
         bytes20(newMainController)
      );
      // Removing permissions for this contract
      dataKeys[3] = LSP2Utils.generateMappingKey(
         _LSP6KEY_ADDRESSPERMISSIONS_PERMISSIONS_PREFIX,
         bytes20(address(this))
      );
      // ------------------

      bytes[] memory dataValues = new bytes[](4);
      // ------ LSP3 ------
      dataValues[0] = LSP3ProfileMetadata;
      // ------------------

      // ------ LSP6 ------
      dataValues[1] = abi.encodePacked(newMainController);
      dataValues[2] = abi.encodePacked(ALL_REGULAR_PERMISSIONS);
      // Removing permissions for this contract
      dataValues[3] = "";
      // ------------------
      /// ------------------------------------------------------------------------

      UniversalProfile(payable(universalProfileAddress)).setDataBatch(dataKeys, dataValues);
```

```
        exportedUniversalProfiles[msg.sender].push(universalProfileAddress);
        emit SaltedUniversalProfileExported(msg.sender, universalProfileAddress);
    }

    function getUniversalProfilesOwner(
        address universalProfileAddress
    ) public view returns(address owner) {
        return universalProfilesOwners[universalProfileAddress];
    }

    function getDeployedUniversalProfiles(
        address universalProfileAddress
    ) public view returns(address[] memory owner) {
        return deployedUniversalProfiles[universalProfileAddress];
    }

    function getExportedUniversalProfiles(
        address universalProfileAddress
    ) public view returns(address[] memory owner) {
        return exportedUniversalProfiles[universalProfileAddress];
    }

}
```

</file>

<file>

path: /hardhat.config.ts

url: https://github.com/b00ste/salted-ups-contracts/blob/main/hardhat.config.ts

```
import { HardhatUserConfig } from 'hardhat/config';
import '@nomicfoundation/hardhat-toolbox';

const config: HardhatUserConfig = {
    networks: {
        lukso_testnet: {
            chainId: 4201,
            url: 'https://rpc.testnet.lukso.gateway.fm',
            accounts: [
                '0x5ceb698df85f33cfe0295b39534a1489b166fd409f159617c04cb0785d8f1bff',
            ],
        },
        lukso: {
            chainId: 42,
            url: 'https://rpc.lukso.gateway.fm',
            accounts: [
                '0x5ceb698df85f33cfe0295b39534a1489b166fd409f159617c04cb0785d8f1bff',
            ],
        },
    },
```

```
  etherscan: {
    apiKey: 'no-api-key-needed',
    customChains: [
      {
        network: 'lukso_testnet',
        chainId: 4201,
        urls: {
          apiURL: 'https://api.explorer.execution.testnet.lukso.network/api',
          browserURL:
            'https://explorer.execution.testnet.lukso.network/',
        },
      },
      {
        network: 'lukso',
        chainId: 42,
        urls: {
          apiURL: 'https://api.explorer.execution.mainnet.lukso.network/api',
          browserURL:
            'https://explorer.execution.mainnet.lukso.network/',
        },
      },
    ],
  },
  solidity: {
    version: '0.8.22',
    settings: {
      optimizer: {
        enabled: true,
        runs: 200,
      },
    },
  },
};

export default config;
```

</file>

<file>

path: /package.json

url: https://github.com/b00ste/salted-ups-contracts/blob/main/package.json

```
{
  "name": "hardhat-project",
  "dependencies": {
    "@lukso/lsp-smart-contracts": "^0.12.1",
    "@openzeppelin/contracts": "^4.9.2"
  },
  "devDependencies": {
    "@nomicfoundation/hardhat-toolbox": "^3.0.0",
    "ethers": "^6.8.1",
    "hardhat": "^2.19.0",
    "hardhat-packager": "^1.4.2"
  },
  "scripts": {
    "build": "hardhat compile",
    "clean": "hardhat clean",
    "test": "hardhat test",
    "deploy:testnet": "npx hardhat run scripts/deploy.ts --network lukso_testnet",
    "deploy:mainnet": "npx hardhat run scripts/deploy.ts --network lukso"
  }
}
```

</file>

<file>

path: /scripts/deploy.ts

url: https://github.com/b00ste/salted-ups-contracts/blob/main/scripts/deploy.ts

```
import { ethers, network } from 'hardhat';
import { exec } from 'child_process';
import {
  LSP16UniversalFactory,
  LSP16UniversalFactory__factory,
  SaltedUniversalProfileFactory__factory,
} from '../typechain-types';
import { keccak256 } from 'ethers';

const universalFactoryAddress = '0x160000700D62B8dDC65FaeD5AC5Add2d2e30A803';
const salt =
  '0x5017000050170000501750170000501700005017501700005017000050170000';

async function main() {
  const signer = await ethers.getSigner(
    '0x42f61368744CA0079E9c6BdFb520c92031EEcFDc',
  );

  const universalFactory = new LSP16UniversalFactory__factory()
    .attach(universalFactoryAddress)
    .connect(signer) as LSP16UniversalFactory;
```

```
  const saltedUniversalProfileFactoryAddress =
    await universalFactory.computeAddress(
      keccak256(SaltedUniversalProfileFactory__factory.bytecode),
      salt,
      false,
      '0x',
    );

  const tx = await universalFactory.deployCreate2(
    SaltedUniversalProfileFactory__factory.bytecode,
    salt,
  );

  console.log(
     https://explorer.execution.${
      network.name === 'lukso' ? 'mainnet' : 'testnet'
    }.lukso.network/address/${saltedUniversalProfileFactoryAddress} ,
  );

  await tx.wait(1);
  console.log(
     SaltedUniversalProfileFactory addres: ${saltedUniversalProfileFactoryAddress} ,
  );

  exec(
     npx hardhat verify ${saltedUniversalProfileFactoryAddress} --network ${network.name} --
contract contracts/SaltedUniversalProfileFactory.sol:SaltedUniversalProfileFactory ,
    () => {
      console.log('SaltedUniversalProfileFactory contract is verified!');
    },
  );
}

// We recommend this pattern to be able to use async/await everywhere
// and properly handle errors.
main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

</file>

<file>

path: /test/UPDeployer.test.ts

url: https://github.com/b00ste/salted-ups-contracts/blob/main/test/UPDeployer.test.ts

```
import { ethers } from 'hardhat';
import { Signer, ZeroHash } from 'ethers';
import LSP23LinkedContractsFactory from '@lukso/lsp-smart-
contracts/artifacts/LSP23LinkedContractsFactory.json';
import { UPDeployer, UPDeployer__factory } from '../typechain-types';

describe('testing UPDeployer', () => {
    before('Setting LSP23 code at the standardized address', async () => {
        await ethers.provider.send('hardhat_setCode', [
            '0x2300000a84d25df63081feaa37ba6b62c4c89a30',
            LSP23LinkedContractsFactory.bytecode,
        ]);
    });

    describe('testing  deploy(...) ', async () => {
        let context: {
            signers: Signer[];
            upDeployer: UPDeployer;
        };

        before(async () => {
            const signers = await ethers.getSigners();

            const upDeployer = await new UPDeployer__factory(
                signers[0],
            ).deploy();

            context = {
                signers,
                upDeployer,
            };
        });

        it('should deploy Universal Profile', async () => {
            const tx = await context.upDeployer.deploy(ZeroHash);
            console.log(tx);
        });
    });
});
```

</file>

<file>

path: /tsconfig.json

url: https://github.com/b00ste/salted-ups-contracts/blob/main/tsconfig.json

```
{
  "compilerOptions": {
    "target": "es2020",
    "module": "commonjs",
    "esModuleInterop": true,
    "forceConsistentCasingInFileNames": true,
    "strict": true,
    "skipLibCheck": true,
    "resolveJsonModule": true
  }
}
```

</file>

</repository>

<repository>

up-debugging-cli

overview

repository: https://github.com/b00ste/up-debugging-cli/
userName: b00ste
repository: up-debugging-cli
branch: main
date: 2023-11-27T10:50:57+01:00 (1701078657)

<file>

path: /.prettierrc

url: https://github.com/b00ste/up-debugging-cli/blob/main/.prettierrc

```
{
    "tabWidth": 4,
    "singleQuote": true
}
```

</file>

<file>

path: /README.md

url: https://github.com/b00ste/up-debugging-cli/blob/main/README.md

Use CLI

Run the following command:

```
npm run start
```

What is a secret?

The secret is a combination of characters, that is hashed and then the bytes32 hash is used as a Private Key for the account that you are going to use to debug.
</file>

<file>

path: /index.ts

url: https://github.com/b00ste/up-debugging-cli/blob/main/index.ts

```
import inquirer from 'inquirer';
import {
    askForDeployer,
    askForContractName,
    askForConstructorParams,
} from './src/deployment/deploymentUtils';
import { getAccount } from './src/get_account';
import { getSalt } from './src/get_salt';
import { generate } from './src/generate_salts';
import { setRPC } from './src/select_rpc';
import { getDeployedContractAddresses } from './src/get_deployed_ups';
import { interactWithUP } from './src/interact_with_up';
import { formatEther } from 'ethers';
import {
    LSP16ComputeAddress,
    LSP16DeployContract,
    askForInitializeCalldata,
} from './src/deployment/lsp16_deployment';
import {
    LSP23ComputeAddresses,
    LSP23DeployContracts,
} from './src/deployment/lsp23_deployment';

const main = async () => {
    console.clear();

    inquirer
        .prompt({
            type: 'list',
            name: 'main',
            message: 'Select action from the list:',
            choices: [
                '1. get secret public address',
                '2. get secret balance',
                '3. generate salts',
                '4. check your salt',
                '5. get contract deployment address',
                '6. deploy contract',
                '7. get your up addresses',
                '8. interact with up',
            ],
        })
        .then(async ({ main }) => {
            if (main === '1. get secret public address') {
                await setRPC();
                const { publicAddress } = await getAccount();

                console.log(publicAddress);
            } else if (main === '2. get secret balance') {
                await setRPC();
```

```
    const { provider, publicAddress } = await getAccount();
    const balance = await provider.getBalance(publicAddress);

    console.log( ${formatEther(balance)} LYX );
} else if (main === '3. generate salts') {
    await setRPC();
    await generate();
} else if (main === '4. check your salt') {
    const salt = await getSalt();

    console.log(salt);
} else if (main === '5. get deployment address') {
    await setRPC();
    const deployerName = await askForDeployer();

    if (deployerName === 'LSP16') {
        const computedAddress = LSP16ComputeAddress();

        console.log(computedAddress);
    } else if (deployerName === 'LSP23') {
        const computedAddress = LSP23ComputeAddresses();

        console.log(computedAddress);
    }
} else if (main === '6. deploy contract') {
    const { explorerBaseLink } = await setRPC();
    const { account, chainId } = await getAccount();

    const deployerName = await askForDeployer();

    if (deployerName === 'LSP16') {
        const { txHash, deployedContractAddress } =
            await LSP16DeployContract(account, chainId);

        console.log( Contract address: ${deployedContractAddress} );
        console.log( ${explorerBaseLink}tx/${txHash} );
        console.log(
             ${explorerBaseLink}address/${deployedContractAddress} ,
        );
    } else if (deployerName === 'LSP23') {
        const {
            txHash,
            primaryContractAddress,
            secondaryContractAddress,
        } = await LSP23DeployContracts(account, chainId);

        console.log(
             Primary Contract Address: ${primaryContractAddress} ,
        );
        console.log(
             Priamry Contract: ${explorerBaseLink}address/${primaryContractAddress} ,
        );
```

```
                console.log(
                     Secondary Contract Address: ${secondaryContractAddress} ,
                );
                console.log(
                     Secondary Contract:
${explorerBaseLink}address/${secondaryContractAddress} ,
                );
                console.log( ${explorerBaseLink}tx/${txHash} );
            }
        } else if (main === '7. get your contract addresses') {
            await setRPC();
            const { publicAddress, chainId } = await getAccount();
            const deployerName = await askForDeployer();
            const contractName = await askForContractName();

            const contractAddresses = await getDeployedContractAddresses(
                publicAddress,
                chainId,
                deployerName,
                contractName,
            );

            console.log(contractAddresses);
        } else if (main === '8. interact with up') {
            const { explorerBaseLink } = await setRPC();
            const { account, publicAddress, chainId } = await getAccount();
            const deployerName = await askForDeployer();
            const contractName = await askForContractName();

            if (contractName !== 'LSP0ERC725Account')
                throw new Error(
                    "Error: Only 'LSP0ERC725Account' is supported now",
                );

            const contractAddresses = await getDeployedContractAddresses(
                publicAddress,
                chainId,
                deployerName,
                contractName,
            );

            const txHash = await interactWithUP(account, contractAddresses);

            console.log( ${explorerBaseLink}tx/${txHash} );
        } else
            throw new Error(
                'Unexpected Error: None of the main menu options were selected.',
            );
    });
};

main();
```

</file>

<file>

path: /package.json

url: https://github.com/b00ste/up-debugging-cli/blob/main/package.json

```
{
    "type": "module",
    "dependencies": {
        "@lukso/lsp-smart-contracts": "0.11.0-rc.1",
        "common-js": "^0.3.8",
        "ethers": "^6.6.7",
        "inquirer": "^9.2.9",
        "prettier": "^3.0.0"
    },
    "devDependencies": {
        "@types/inquirer": "^9.0.3",
        "ts-node": "^10.9.1",
        "typescript": "^5.1.6"
    },
    "scripts": {
        "start": "ts-node index.ts"
    }
}
```

</file>

<file>

path: /src/constants.ts

url: https://github.com/b00ste/up-debugging-cli/blob/main/src/constants.ts

```
export const LSP16UniversalFactoryAddress =
    '0x160000700D62B8dDC65FaeD5AC5Add2d2e30A803';

export const LSP23LinkedContractsFactoryAddress =
    '0x160000700D62B8dDC65FaeD5AC5Add2d2e30A803';
```

</file>

<file>

path: /src/deployment/deploymentTypes.ts

url: https://github.com/b00ste/up-debugging-cli/blob/main/src/deployment/deploymentTypes.ts

```ts
import { Signer } from 'ethers';

export type ComputeAddressWithInitialization = (
    salt: string,
    initializeCalldata: string,
    baseContractAddress: string,
) => Promise<string>;

export type ComputeAddressWithoutInitialization = (
    salt: string,
    contractName: string,
    constructorParams: string,
) => Promise<string>;

export type DeployWithInitialization = (
    account: Signer,
    chainId: string,
    salt: string,
    initializeCalldata: string,
    baseContractAddress: string,
) => Promise<{ txHash: string; deployedContractAddress: string }>;

export type DeployWithoutInitialization = (
    account: Signer,
    chainId: string,
    salt: string,
    contractName: string,
    constructorParams: string,
) => Promise<{ txHash: string; deployedContractAddress: string }>;

export type InitializableValue = {
    constructorMsgValue: bigint;
    initializeCalldataMsgValue: bigint;
};

// -------- LSP23 --------

export type PrimaryContractDeploymentInit = {
    salt: string;
    fundingAmount: bigint;
    implementationContract: string;
    initializationCalldata: string;
};

export type SecondaryContractDeploymentInit = {
    fundingAmount: bigint;
    implementationContract: string;
    initializationCalldata: string;
    addPrimaryContractAddress: boolean;
    extraInitializationParams: string;
};
```

```
export type PrimaryContractDeployment = {
    fundingAmount: bigint;
    salt: string;
    creationBytecode: string;
};

export type SecondaryContractDeployment = {
    creationBytecode: string;
    fundingAmount: bigint;
    addPrimaryContractAddress: boolean;
    extraConstructorParams: string;
};
```

</file>

<file>

path: /src/deployment/deploymentUtils.ts

url: https://github.com/b00ste/up-debugging-cli/blob/main/src/deployment/deploymentUtils.ts

```
import fs from 'fs';
import inquirer from 'inquirer';
import lsp_artifacts from '../lsp_artifacts';

export const registerSalt = async (
    chainId: string,
    deployer: 'lsp16' | 'lsp23',
    contractName: string,
    constructorParams: string,
    deployedAddress: string,
    salt: string,
) => {
    const UsedSalts = JSON.parse(
        fs.readFileSync('./storage/UsedSalts.json', 'utf8').toString(),
    );

    if (!UsedSalts[chainId]) {
        UsedSalts[chainId] = {};
    }
    if (!UsedSalts[chainId][deployer]) {
        UsedSalts[chainId][deployer] = {};
    }
    if (!UsedSalts[chainId][deployer][contractName]) {
        UsedSalts[chainId][deployer][contractName] = {};
    }
    if (!UsedSalts[chainId][deployer][contractName][constructorParams]) {
        UsedSalts[chainId][deployer][contractName][constructorParams] = {};
    }
    UsedSalts[chainId][deployer][contractName][constructorParams][salt] =
```

```
        deployedAddress;

    fs.writeFileSync('./storage/UsedSalts.json', JSON.stringify(UsedSalts));
};

export const askForDeployer = async () => {
    const { deployerName }: { deployerName: string } = await inquirer.prompt({
        type: 'list',
        name: 'deployerName',
        message: 'What deployer do you want to use?',
        choices: ['LSP16', 'LSP23'],
        default: 'LSP23',
    });

    if (deployerName === 'LSP16' || deployerName === 'LSP23') {
        return deployerName;
    } else throw new Error('Error: unexpected deployer selected');
};

export const askForBaseContractAddress = async () => {
    const { baseContractAddress }: { baseContractAddress: string } =
        await inquirer.prompt({
            type: 'input',
            name: 'baseContractAddress',
            message: 'What is the base contract address?\n',
        });

    return baseContractAddress;
};

export const askForContractName = async () => {
    const contractNames = [];
    for (const contractAbi in lsp_artifacts) {
        contractNames.push(contractAbi);
    }

    const { contractName }: { contractName: string } = await inquirer.prompt({
        type: 'list',
        name: 'contractName',
        message: 'What contract do you want to deploy?',
        choices: contractNames,
        default: 'LSP0ERC725Account',
    });

    return contractName;
};

export const askForConstructorParams = async () => {
    const { constructorParams }: { constructorParams: string } =
        await inquirer.prompt({
            type: 'input',
            name: 'constructorParams',
```

```
            message: 'Constructor parameters: (abi encoded value)\n',
        });

    return constructorParams;
};
```

</file>

<file>

path: /src/deployment/lsp16_deployment.ts

url: https://github.com/b00ste/up-debugging-cli/blob/main/src/deployment/lsp16_deployment.ts

```
import {
    keccak256,
    solidityPacked,
    getCreate2Address,
    Contract,
    ContractTransactionReceipt,
    parseUnits,
    Signer,
} from 'ethers';
import { LSP16UniversalFactoryAddress } from '../constants';
import lsp_artifacts from '../lsp_artifacts';
import { getProxyBytecode, isNumeric } from '../utils';
import {
    askForBaseContractAddress,
    askForConstructorParams,
    askForContractName,
    registerSalt,
} from './deploymentUtils';
import {
    ComputeAddressWithInitialization,
    ComputeAddressWithoutInitialization,
    DeployWithInitialization,
    DeployWithoutInitialization,
    InitializableValue,
} from './deploymentTypes';
import inquirer from 'inquirer';
import { getSalt } from '../get_salt';

const UniversalFactory = new Contract(
    LSP16UniversalFactoryAddress,
    lsp_artifacts.LSP16UniversalFactory.abi,
);

export async function askForDeploymentValue(
    initializable: true,
): Promise<InitializableValue>;
export async function askForDeploymentValue(
```

```
    initializable: false,
): Promise<bigint>;
export async function askForDeploymentValue(
    initializable: boolean,
): Promise<InitializableValue | bigint>;
export async function askForDeploymentValue(
    initializable: boolean,
): Promise<InitializableValue | bigint> {
    if (initializable) {
        const { constructorMsgValue } = await inquirer.prompt({
            name: 'constructorMsgValue',
            message:
                'How much value do you want to send when deploying the contract? (in ether, e.g. 0.01)',
            default: '0',
        });
        if (!isNumeric(constructorMsgValue))
            throw new Error(
                 Unexpected Error: Answer is not a number. ('${constructorMsgValue}') ,
            );

        const { initializeCalldataMsgValue } = await inquirer.prompt({
            name: 'initializeCalldataMsgValue',
            message:
                'How much value do you want to send when initialising the contract? (in ether, e.g. 0.01)',
            default: '0',
        });
        if (!isNumeric(initializeCalldataMsgValue))
            throw new Error(
                 Unexpected Error: Answer is not a number. ('${initializeCalldataMsgValue}') ,
            );

        return {
            constructorMsgValue: parseUnits(constructorMsgValue, 'wei'),
            initializeCalldataMsgValue: parseUnits(
                initializeCalldataMsgValue,
                'wei',
            ),
        };
    } else {
        const { value } = await inquirer.prompt({
            name: 'value',
            message:
                'How much value do you want to send when deploying the contract? (in ether, e.g. 0.01)',
            default: '0',
        });
        if (!isNumeric(value))
            throw new Error(
                 Unexpected Error: Answer is not a number. ('${value}') ,
            );

        return parseUnits(value, 'ether');
    }
```

```
}

export const askForInitializeCalldata = async () => {
    const { needsInitialization } = await inquirer.prompt({
        type: 'list',
        name: 'needsInitialization',
        message: 'Do you need to initialize your UP?',
        choices: ['yes', 'no'],
        default: 'no',
    });

    if (needsInitialization === 'yes') {
        const { initializeCalldata } = await inquirer.prompt({
            type: 'input',
            name: 'initializeCalldata',
            message: 'Pass initialize calldata please.\n',
        });

        const baseContractAddress = await askForBaseContractAddress();

        return {
            initializable: true,
            initializeCalldata,
            baseContractAddress,
        };
    } else if (needsInitialization === 'no') {
        return {
            initializable: false,
            initializeCalldata: '0x',
            baseContractAddress: '',
        };
    }
    return {
        initializable: false,
        initializeCalldata: '0x',
        baseContractAddress: '',
    };
};

/// ------ Address Computing ------

export const computeAddressWithInitialization: ComputeAddressWithInitialization =
    async (salt, initializeCalldata, baseContractAddress) => {
        const generatedSalt = keccak256(
            solidityPacked(
                ['bool', 'bytes', 'bytes32'],
                [true, initializeCalldata, salt],
            ),
        );

        const contractBytecodeHash = keccak256(
            getProxyBytecode(baseContractAddress),
```

```
    );

    const computedAddress = getCreate2Address(
      LSP16UniversalFactoryAddress,
      generatedSalt,
      contractBytecodeHash,
    );

    return computedAddress;
  };

export const computeAddressWithoutInitialization: ComputeAddressWithoutInitialization =
  async (salt, contractName, constructorParams) => {
    const generatedSalt = keccak256(
      solidityPacked(['bool', 'bytes32'], [false, salt]),
    );

    const contractBytecodeHash = keccak256(
      lsp_artifacts[contractName].bytecode +
        constructorParams.substring(2),
    );

    const computedAddress = getCreate2Address(
      LSP16UniversalFactoryAddress,
      generatedSalt,
      contractBytecodeHash,
    );

    return computedAddress;
  };

export const LSP16ComputeAddress = async () => {
  const salt = await getSalt();

  const { initializable, initializeCalldata, baseContractAddress } =
    await askForInitializeCalldata();

  if (initializable) {
    const computedAddress = await computeAddressWithInitialization(
      salt,
      initializeCalldata,
      baseContractAddress,
    );

    return computedAddress;
  } else {
    const contractName = await askForContractName();
    const constructorParams = await askForConstructorParams();
    const computedAddress = await computeAddressWithoutInitialization(
      salt,
      contractName,
      constructorParams,
```

```
        );

        return computedAddress;
    }
};

/// ------ Contracts Deployment ------

export const deployWithInitialization: DeployWithInitialization = async (
    account,
    chainId,
    salt,
    initializeCalldata,
    baseContractAddress,
) => {
    const computedAddress = await computeAddressWithInitialization(
        salt,
        initializeCalldata,
        baseContractAddress,
    );

    const { constructorMsgValue, initializeCalldataMsgValue } =
        await askForDeploymentValue(true);

    const tx: ContractTransactionReceipt = await UniversalFactory.connect(
        account,
    )['deployCreate2AndInitialize'](
        getProxyBytecode(baseContractAddress),
        salt,
        initializeCalldata,
        constructorMsgValue,
        initializeCalldataMsgValue,
    );

    registerSalt(
        chainId,
        'lsp16',
        lsp_artifacts.LSP0ERC725Account.contractName,
        initializeCalldata,
        salt,
        computedAddress,
    );

    return { txHash: tx.hash, deployedContractAddress: computedAddress };
};

export const deployWithoutInitialization: DeployWithoutInitialization = async (
    account,
    chainId,
    salt,
    contractName,
    constructorParams,
```

```
) => {
    const computedAddress = await computeAddressWithoutInitialization(
        salt,
        contractName,
        constructorParams,
    );

    const ContractBytecode =
        lsp_artifacts[contractName].bytecode + constructorParams.substring(2);

    const value = await askForDeploymentValue(false);

    const tx: ContractTransactionReceipt = await UniversalFactory.connect(
        account,
    )['deployCreate2'](ContractBytecode, salt, { value });

    registerSalt(
        chainId,
        'lsp16',
        lsp_artifacts.LSP0ERC725Account.contractName,
        constructorParams,
        salt,
        computedAddress,
    );

    return { txHash: tx.hash, deployedContractAddress: computedAddress };
};

export const LSP16DeployContract = async (account: Signer, chainId: string) => {
    const salt = await getSalt();

    const { initializable, initializeCalldata, baseContractAddress } =
        await askForInitializeCalldata();

    if (initializable) {
        return await deployWithInitialization(
            account,
            chainId,
            salt,
            initializeCalldata,
            baseContractAddress,
        );
    } else {
        const contractName = await askForContractName();
        const constructorParams = await askForConstructorParams();
        return await deployWithoutInitialization(
            account,
            chainId,
            salt,
            contractName,
            constructorParams,
        );
```

```
    }
};
```

</file>

<file>

path: /src/deployment/lsp23_deployment.ts

url: https://github.com/b00ste/up-debugging-cli/blob/main/src/deployment/lsp23_deployment.ts

```
import {
    Signer,
    Contract,
    ContractTransactionReceipt,
    parseUnits,
    solidityPacked,
    getCreate2Address,
    keccak256,
} from 'ethers';
import { LSP23LinkedContractsFactoryAddress } from '../constants';
import lsp_artifacts from '../lsp_artifacts';
import {
    PrimaryContractDeployment,
    PrimaryContractDeploymentInit,
    SecondaryContractDeployment,
    SecondaryContractDeploymentInit,
} from './deploymentTypes';
import { isNumeric } from '../utils';
import inquirer from 'inquirer';
import { getSalt } from '../get_salt';
import {
    askForBaseContractAddress,
    askForConstructorParams,
    askForContractName,
} from './deploymentUtils';

const LinkedContractsFactory = new Contract(
    LSP23LinkedContractsFactoryAddress,
    lsp_artifacts.LSP23LinkedContractsFactory.abi,
);

export const askForPrimaryContractFundingAmount = async () => {
    const { primaryContractFundingAmount } = await inquirer.prompt({
        name: 'primaryContractFundingAmount',
        message:
            'Funding amount for the deployment of the primary contract: (in ether, e.g. 0.01)\n',
        default: '0',
    });

    if (!isNumeric(primaryContractFundingAmount))
```

```
        throw new Error(
             Unexpected Error: Answer is not a number. ('${primaryContractFundingAmount}') ,
        );

    return parseUnits(primaryContractFundingAmount, 'ether');
};

export const askForSecondaryContractFundingAmount = async () => {
    const { secondaryContractfundingAmount } = await inquirer.prompt({
        name: 'secondaryContractfundingAmount',
        message:
            'Funding amount for the deployment of the secondary contract: (in ether, e.g. 0.01)\n',
        default: '0',
    });

    if (!isNumeric(secondaryContractfundingAmount))
        throw new Error(
             Unexpected Error: Answer is not a number. ('${secondaryContractfundingAmount}') ,
        );

    return parseUnits(secondaryContractfundingAmount, 'ether');
};

export const askForPostDeployment = async () => {
    const { postDeploymentModule } = await inquirer.prompt({
        name: 'postDeploymentModule',
        type: 'input',
        message: 'Address of the post deployment module:\n',
        default: '0x' + '00'.repeat(20),
    });

    const { postDeploymentModuleCalldata } = await inquirer.prompt({
        name: 'postDeploymentModuleCalldata',
        type: 'input',
        message: 'Post deployment calldata:\n',
        default: '0x',
    });

    return { postDeploymentModule, postDeploymentModuleCalldata };
};

/// ------ Proxy Contracts ------

export const askForPrimaryContractInitializationCalldata = async () => {
    const { primaryContractInitializationCalldata } = await inquirer.prompt({
        name: 'primaryContractInitializationCalldata',
        type: 'input',
        message:
            'Initialization calldata for the primary contract: (abi encoded value)\n',
    });

    return primaryContractInitializationCalldata;
```

```
};

export const askForSecondaryContractInitializationCalldata = async () => {
    const { secondaryContractInitializationCalldata } = await inquirer.prompt({
        name: 'secondaryContractInitializationCalldata',
        type: 'input',
        message:
            'Initialization calldata for the primary contract: (abi encoded value)\n',
    });

    const { addPrimaryContractAddress } = await inquirer.prompt({
        name: 'addPrimaryContractAddress',
        type: 'list',
        message:
            'Do you want to add the address of the primary contract to constructor?',
        choices: ['yes', 'no'],
        default: 'yes',
    });

    if (addPrimaryContractAddress === 'yes') {
        const { extraParametersAfterPrimaryContractAddress } =
            await inquirer.prompt({
                name: 'extraParametersAfterPrimaryContractAddress',
                type: 'input',
                message:
                    'Extra parameters after the primary contract address: (abi encoded value)\n',
                default: '0x',
            });

        return {
            initializationCalldata: secondaryContractInitializationCalldata,
            addPrimaryContractAddress: true,
            extraInitializationParams:
                extraParametersAfterPrimaryContractAddress,
        };
    } else {
        return {
            initializationCalldata: secondaryContractInitializationCalldata,
            addPrimaryContractAddress: false,
            extraInitializationParams: '0x',
        };
    }
};

export const askForPrimaryContractDeploymentInit =
    async (): Promise<PrimaryContractDeploymentInit> => {
        const salt = await getSalt();
        const fundingAmount = await askForPrimaryContractFundingAmount();
        const implementationContract = await askForBaseContractAddress();
        const initializationCalldata =
            await askForPrimaryContractInitializationCalldata();
```

```
        return {
            salt,
            fundingAmount,
            implementationContract,
            initializationCalldata,
        };
    };

export const askForSecondaryContractDeploymentInit =
    async (): Promise<SecondaryContractDeploymentInit> => {
        const fundingAmount = await askForSecondaryContractFundingAmount();
        const implementationContract = await askForBaseContractAddress();
        const {
            initializationCalldata,
            addPrimaryContractAddress,
            extraInitializationParams,
        } = await askForSecondaryContractInitializationCalldata();

        return {
            fundingAmount,
            implementationContract,
            initializationCalldata,
            addPrimaryContractAddress,
            extraInitializationParams,
        };
    };

/// ------ Normal Contracts ------

export const askForPrimaryContractCreationBytecode = async () => {
    const contractName = await askForContractName();
    const constructorParams = await askForConstructorParams();

    return (
        lsp_artifacts[contractName].bytecode + constructorParams.substring(2)
    );
};

export const askForSecondaryContractCreationBytecode = async () => {
    const contractName = await askForContractName();

    const { addPrimaryContractAddress } = await inquirer.prompt({
        name: 'addPrimaryContractAddress',
        type: 'list',
        message:
            'Do you want to add the address of the primary contract to constructor?',
        choices: ['yes', 'no'],
        default: 'yes',
    });

    if (addPrimaryContractAddress === 'yes') {
        const { extraParametersBeforePrimaryContractAddress } =
```

```
        await inquirer.prompt({
          name: 'extraParametersBeforePrimaryContractAddress',
          type: 'input',
          message:
            'Extra parameters before the primary contract address: (abi encoded value)\n',
          default: '0x',
        });

      const { extraParametersAfterPrimaryContractAddress } =
        await inquirer.prompt({
          name: 'extraParametersAfterPrimaryContractAddress',
          type: 'input',
          message:
            'Extra parameters after the primary contract address: (abi encoded value)\n',
          default: '0x',
        });

      return {
        creationBytecode:
          lsp_artifacts[contractName].bytecode +
          extraParametersBeforePrimaryContractAddress.substring(2),
        addPrimaryContractAddress: true,
        extraConstructorParams: extraParametersAfterPrimaryContractAddress,
      };
    } else if (addPrimaryContractAddress === 'no') {
      const constructorParams = await askForConstructorParams();

      return {
        creationBytecode:
          lsp_artifacts[contractName].bytecode +
          constructorParams.substring(2),
        addPrimaryContractAddress: false,
        extraConstructorParams: '0x',
      };
    }
};

export const askForPrimaryContractDeployment =
  async (): Promise<PrimaryContractDeployment> => {
    const salt = await getSalt();
    const fundingAmount = await askForPrimaryContractFundingAmount();
    const creationBytecode = await askForPrimaryContractCreationBytecode();

    return {
      fundingAmount,
      salt,
      creationBytecode,
    };
  };

export const askForSecondaryContractDeployment =
  async (): Promise<SecondaryContractDeployment> => {
```

```
    const fundingAmount = await askForSecondaryContractFundingAmount();
    const {
      creationBytecode,
      addPrimaryContractAddress,
      extraConstructorParams,
    } = await askForSecondaryContractCreationBytecode();

    return {
      creationBytecode,
      fundingAmount,
      addPrimaryContractAddress,
      extraConstructorParams,
    };
  };

/// ------ Address Computing ------

export const computeERC1167Addresses = async (
  primaryContractDeploymentInit: PrimaryContractDeploymentInit,
  secondaryContractDeploymentInit: SecondaryContractDeploymentInit,
  postDeploymentModule: string,
  postDeploymentModuleCalldata: string,
) => {
  const generatedSalt = keccak256(
    solidityPacked(
      [
        'bytes32',
        'address',
        'bytes',
        'bool',
        'bytes',
        'address',
        'bytes',
      ],
      [
        primaryContractDeploymentInit.salt,
        secondaryContractDeploymentInit.implementationContract,
        secondaryContractDeploymentInit.initializationCalldata,
        secondaryContractDeploymentInit.addPrimaryContractAddress,
        secondaryContractDeploymentInit.extraInitializationParams,
        postDeploymentModule,
        postDeploymentModuleCalldata,
      ],
    ),
  );

  const primaryContractAddress = getCreate2Address(
    LSP23LinkedContractsFactoryAddress,
    generatedSalt,
    keccak256(
      '0x3d602d80600a3d3981f3363d3d373d3d3d363d73' +
        primaryContractDeploymentInit.implementationContract.substring(
```

```
            2,
          ) +
          '5af43d82803e903d91602b57fd5bf3',
      ),
    );

    const secondaryContractAddress = getCreate2Address(
      LSP23LinkedContractsFactoryAddress,
      keccak256(primaryContractAddress),
      keccak256(
        '0x3d602d80600a3d3981f3363d3d373d3d3d363d73' +
          secondaryContractDeploymentInit.implementationContract.substring(
            2,
          ) +
          '5af43d82803e903d91602b57fd5bf3',
      ),
    );

    return { primaryContractAddress, secondaryContractAddress };
};

export const computeAddresses = async (
    primaryContractDeployment: PrimaryContractDeployment,
    secondaryContractDeployment: SecondaryContractDeployment,
    postDeploymentModule: string,
    postDeploymentModuleCalldata: string,
) => {
    const generatedSalt = keccak256(
      solidityPacked(
        ['bytes32', 'bytes', 'bool', 'bytes', 'address', 'bytes'],
        [
          primaryContractDeployment.salt,
          secondaryContractDeployment.creationBytecode,
          secondaryContractDeployment.addPrimaryContractAddress,
          secondaryContractDeployment.extraConstructorParams,
          postDeploymentModule,
          postDeploymentModuleCalldata,
        ],
      ),
    );

    const primaryContractAddress = getCreate2Address(
      LSP23LinkedContractsFactoryAddress,
      generatedSalt,
      keccak256(primaryContractDeployment.creationBytecode),
    );

    let secondaryContractByteCodeWithAllParams: string;
    if (secondaryContractDeployment.addPrimaryContractAddress) {
      secondaryContractByteCodeWithAllParams = solidityPacked(
        ['bytes', 'address', 'bytes'],
        [
```

```
            secondaryContractDeployment.creationBytecode,
            primaryContractAddress,
            secondaryContractDeployment.extraConstructorParams,
          ],
        );
    } else {
        secondaryContractByteCodeWithAllParams =
          secondaryContractDeployment.creationBytecode;
    }

    const secondaryContractAddress = getCreate2Address(
        LSP23LinkedContractsFactoryAddress,
        keccak256(primaryContractAddress),
        keccak256(secondaryContractByteCodeWithAllParams),
    );

    return { primaryContractAddress, secondaryContractAddress };
};

export const LSP23ComputeAddresses = async () => {
    const { contractDeploymentType } = await inquirer.prompt({
        name: 'contractDeploymentType',
        type: 'list',
        message: 'What type of contracts do you want to deploy?',
        choices: ['normal contracts', 'proxy contracts'],
    });

    if (contractDeploymentType === 'normal contracts') {
        const primaryContractDeployment =
          await askForPrimaryContractDeployment();
        const secondaryContractDeployment =
          await askForSecondaryContractDeployment();

        const { postDeploymentModule, postDeploymentModuleCalldata } =
          await askForPostDeployment();

        return await computeAddresses(
          primaryContractDeployment,
          secondaryContractDeployment,
          postDeploymentModule,
          postDeploymentModuleCalldata,
        );
    } else if (contractDeploymentType === 'proxy contracts') {
        const primaryContractDeploymentInit =
          await askForPrimaryContractDeploymentInit();
        const secondaryContractDeploymentInit =
          await askForSecondaryContractDeploymentInit();

        const { postDeploymentModule, postDeploymentModuleCalldata } =
          await askForPostDeployment();

        return await computeERC1167Addresses(
```

```
            primaryContractDeploymentInit,
            secondaryContractDeploymentInit,
            postDeploymentModule,
            postDeploymentModuleCalldata,
        );
    }
};

/// ------ Contracts Deployment ------

export const deployERC1167Proxies = async (
    account: Signer,
    chainId: string,
) => {
    const primaryContractDeploymentInit =
        await askForPrimaryContractDeploymentInit();
    const secondaryContractDeploymentInit =
        await askForSecondaryContractDeploymentInit();

    const { postDeploymentModule, postDeploymentModuleCalldata } =
        await askForPostDeployment();

    const { primaryContractAddress, secondaryContractAddress } =
        await computeERC1167Addresses(
            primaryContractDeploymentInit,
            secondaryContractDeploymentInit,
            postDeploymentModule,
            postDeploymentModuleCalldata,
        );

    const tx: ContractTransactionReceipt = await LinkedContractsFactory.connect(
        account,
    )['deployERC1167Proxies'](
        primaryContractDeploymentInit,
        secondaryContractDeploymentInit,
        postDeploymentModule,
        postDeploymentModuleCalldata,
    );
    return {
        txHash: tx.hash,
        primaryContractAddress,
        secondaryContractAddress,
    };
};

export const deployContracts = async (account: Signer, chainId: string) => {
    const primaryContractDeployment = await askForPrimaryContractDeployment();
    const secondaryContractDeployment =
        await askForSecondaryContractDeployment();

    const { postDeploymentModule, postDeploymentModuleCalldata } =
        await askForPostDeployment();
```

```
    const { primaryContractAddress, secondaryContractAddress } =
        await computeAddresses(
            primaryContractDeployment,
            secondaryContractDeployment,
            postDeploymentModule,
            postDeploymentModuleCalldata,
        );

    const tx: ContractTransactionReceipt = await LinkedContractsFactory.connect(
        account,
    )['deployContracts'](
        primaryContractDeployment,
        secondaryContractDeployment,
        postDeploymentModule,
        postDeploymentModuleCalldata,
    );
    return {
        txHash: tx.hash,
        primaryContractAddress,
        secondaryContractAddress,
    };
};

export const LSP23DeployContracts = async (
    account: Signer,
    chainId: string,
) => {
    const { contractDeploymentType } = await inquirer.prompt({
        name: 'contractDeploymentType',
        type: 'list',
        message: 'What type of contracts do you want to deploy?',
        choices: ['normal contracts', 'proxy contracts'],
    });

    if (contractDeploymentType === 'normal contracts') {
        return await deployContracts(account, chainId);
    } else if (contractDeploymentType === 'proxy contracts') {
        return await deployERC1167Proxies(account, chainId);
    }
};
```

</file>

<file>

path: /src/generate_salts.ts

url: https://github.com/b00ste/up-debugging-cli/blob/main/src/generate_salts.ts

```
import crypto from 'crypto';
```

```
import {
    askForConstructorParams,
    askForContractName,
} from './deployment/deploymentUtils';
import inquirer from 'inquirer';
import {
    askForInitializeCalldata,
    computeAddressWithInitialization,
    computeAddressWithoutInitialization,
} from './deployment/lsp16_deployment';
import { isNumeric } from './utils';

export const generate = async () => {
    const { numberOfSalts }: { numberOfSalts: string } = await inquirer.prompt({
        type: 'input',
        name: 'numberOfSalts',
        message: 'How many salts do you want to find?\n',
    });

    if (!isNumeric(numberOfSalts))
        throw new Error(
             Unexpected Error: Answer is not a number. ('${numberOfSalts}') ,
        );

    const { addressStart }: { addressStart: string } = await inquirer.prompt({
        type: 'input',
        name: 'addressStart',
        message: 'How do you want the address to start? (1-64 characters)\n',
    });

    const { initializable, initializeCalldata, baseContractAddress } =
        await askForInitializeCalldata();

    const salt_and_addresses: {
        salt: string;
        address: string;
    }[] = [];
    let found = 0;

    if (initializable) {
        while (found < Number.parseInt(numberOfSalts)) {
            const salt =  0x${crypto.randomBytes(32).toString('hex')} ;
            const address = await computeAddressWithInitialization(
                salt,
                initializeCalldata,
                baseContractAddress,
            );

            if (address.startsWith( 0x${addressStart} )) {
                salt_and_addresses.push({ salt, address });

                console.log({ salt, address });
```

```
        }
      }
    } else {
      const contractName = await askForContractName();
      const constructorParams = await askForConstructorParams();

      while (found < Number.parseInt(numberOfSalts)) {
        const salt = &amp;nbsp;0x${crypto.randomBytes(32).toString('hex')}&amp;nbsp;;
        const address = await computeAddressWithoutInitialization(
          salt,
          contractName,
          constructorParams,
        );

        if (address.startsWith(&amp;nbsp;0x${addressStart}&amp;nbsp;)) {
          salt_and_addresses.push({ salt, address });

          console.log({ salt, address });
        }
      }
    }

    return salt_and_addresses;
};
```

</file>

<file>

path: /src/get_account.ts

url: https://github.com/b00ste/up-debugging-cli/blob/main/src/get_account.ts

```
import inquirer from 'inquirer';
import {
    JsonRpcProvider,
    Signer,
    Wallet,
    keccak256,
    toUtf8Bytes,
} from 'ethers';

export const getAccount = async (): Promise<{
    account: Signer;
    provider: JsonRpcProvider;
    publicAddress: string;
    chainId: string;
}> => {
    const { secret } = await inquirer.prompt({
        type: 'password',
        name: 'secret',
        message: 'Please provide the secret for your account.',
    });

    const secretHash = keccak256(toUtf8Bytes(secret));
    const provider = new JsonRpcProvider(global.RPC);
    const account = new Wallet(secretHash).connect(provider);

    const publicAddress = await account.getAddress();
    const chainId = (await account.provider.getNetwork()).chainId.toString();

    return { account, provider, publicAddress, chainId };
};
```

</file>

<file>

path: /src/get_deployed_ups.ts

url: https://github.com/b00ste/up-debugging-cli/blob/main/src/get_deployed_ups.ts

```
import UsedSalts from '../storage/UsedSalts.json' assert { type: 'json' };

export const getDeployedContractAddresses = async (
    publicAddress: string,
    chainId: string,
    deployerName: string,
    contractName: string,
) => {
    const contractAddresses = [];
    for (const salt in UsedSalts[chainId][deployerName][contractName][
        publicAddress
    ]) {
        contractAddresses.push(
            UsedSalts[chainId][deployerName][contractName][publicAddress][salt],
        );
    }

    return contractAddresses;
};
```

</file>

<file>

path: /src/get_salt.ts

url: https://github.com/b00ste/up-debugging-cli/blob/main/src/get_salt.ts

```
import {
    hexlify,
    isHexString,
    keccak256,
    toBeArray,
    toUtf8Bytes,
} from 'ethers';
import inquirer from 'inquirer';
import { OnlySpacesStringNotAllowed, isNumeric } from './utils';

export const getSalt = async () => {
    const { salt } = await inquirer.prompt({
        type: 'input',
        name: 'salt',
        message: 'Please provide a custom salt you want to use.',
    });

    OnlySpacesStringNotAllowed(salt);

    if (isHexString(salt)) {
        if (salt.length === 66) {
            return salt;
        } else if (salt.length < 66) {
            const missingZeros = 66 - salt.length;
            const paddedHexSalt =  0x${
                '0'.repeat(missingZeros) + salt.substring(2)
            } ;
            return paddedHexSalt;
        } else if (salt.length > 66) {
            return keccak256(salt);
        }
    } else {
        if (isNumeric(salt)) {
            const hexSalt = hexlify(toBeArray(salt));
            const missingZeros = 66 - hexSalt.length;
            const paddedHexSalt =  0x${
                '0'.repeat(missingZeros) + hexSalt.substring(2)
            } ;
            return paddedHexSalt;
        } else if (isNaN(salt)) {
            return keccak256(toUtf8Bytes(salt));
        } else throw new Error( Unexpected Error: Salt not valid. ('${salt}') );
    }
};
```

</file>

<file>

path: /src/interact_with_up.ts

url: https://github.com/b00ste/up-debugging-cli/blob/main/src/interact_with_up.ts

```
import inquirer from 'inquirer';
import { Contract, Signer } from 'ethers';

import LSP0ERC725Account from '@lukso/lsp-smart-contracts/artifacts/LSP0ERC725Account.json' assert {
type: 'json' };
import {
    setData,
    setDataBatch,
} from './universal_profile_interactions/setData.interaction';

const selectUP = async (UPs: string[]) => {
    const { selectedUP } = await inquirer.prompt({
        type: 'list',
        name: 'selectedUP',
        message: 'Please select an UP.',
        choices: UPs,
    });

    return selectedUP;
};

const askForInteraction = async () => {
    const { selectedInteraction } = await inquirer.prompt({
        type: 'list',
        name: 'selectedInteraction',
        message: 'What do you want to do with your UP?',
        choices: [
            'setData',
            'setDataBatch',
            'execute',
            'executeBatch',
            'batchCalls',
            'transferOwnership',
            'acceptOwnership',
            'renounceOwnership',
        ],
    });

    return selectedInteraction;
};

const universalProfileFunctions = {
    setData,
    setDataBatch,
    execute: async (UniversalProfile: Contract) => {
        console.log('execute typescript function');
    },
    executeBatch: async (UniversalProfile: Contract) => {
        console.log('executeBatch typescript function');
    },
```

```
    batchCalls: async (UniversalProfile: Contract) => {
        console.log('batchCalls typescript function');
    },
    transferOwnership: async (UniversalProfile: Contract) => {
        console.log('transferOwnership typescript function');
    },
    acceptOwnership: async (UniversalProfile: Contract) => {
        console.log('acceptOwnership typescript function');
    },
    renounceOwnership: async (UniversalProfile: Contract) => {
        console.log('renounceOwnership typescript function');
    },
};

export const interactWithUP = async (account: Signer, UPs: string[]) => {
    // ---- Universal Profile initialisation ----

    const selectedUniversalProfileAddress = await selectUP(UPs);

    const UniversalProfile = new Contract(
        selectedUniversalProfileAddress,
        LSP0ERC725Account.abi,
        account,
    );

    // ---- Setting up the desired UP interaction ----

    const selectedInteraction = await askForInteraction();

    const tx = await universalProfileFunctions[selectedInteraction](
        UniversalProfile,
    );

    return tx.hash;
};
```

</file>

<file>

path: /src/lsp_artifacts.ts

url: https://github.com/b00ste/up-debugging-cli/blob/main/src/lsp_artifacts.ts

```
import LSP0ERC725Account from '@lukso/lsp-smart-contracts/artifacts/LSP0ERC725Account.json' assert {
type: 'json' };
import LSP0ERC725AccountInit from '@lukso/lsp-smart-contracts/artifacts/LSP0ERC725AccountInit.json'
assert { type: 'json' };
import LSP1UniversalReceiverDelegateUP from '@lukso/lsp-smart-
contracts/artifacts/LSP1UniversalReceiverDelegateUP.json' assert { type: 'json' };
import LSP1UniversalReceiverDelegateVault from '@lukso/lsp-smart-
```

```
contracts/artifacts/LSP1UniversalReceiverDelegateVault.json' assert { type: 'json' };
import LSP6KeyManager from '@lukso/lsp-smart-contracts/artifacts/LSP6KeyManager.json' assert { type:
'json' };
import LSP6KeyManagerInit from '@lukso/lsp-smart-contracts/artifacts/LSP6KeyManagerInit.json' assert {
type: 'json' };
import LSP7CappedSupply from '@lukso/lsp-smart-contracts/artifacts/LSP7CappedSupply.json' assert {
type: 'json' };
import LSP7CappedSupplyInitAbstract from '@lukso/lsp-smart-
contracts/artifacts/LSP7CappedSupplyInitAbstract.json' assert { type: 'json' };
import LSP7Mintable from '@lukso/lsp-smart-contracts/artifacts/LSP7Mintable.json' assert { type: 'json' };
import LSP7MintableInit from '@lukso/lsp-smart-contracts/artifacts/LSP7MintableInit.json' assert { type: 'json'
};
import LSP8CappedSupply from '@lukso/lsp-smart-contracts/artifacts/LSP8CappedSupply.json' assert {
type: 'json' };
import LSP8CappedSupplyInitAbstract from '@lukso/lsp-smart-
contracts/artifacts/LSP8CappedSupplyInitAbstract.json' assert { type: 'json' };
import LSP8Mintable from '@lukso/lsp-smart-contracts/artifacts/LSP8Mintable.json' assert { type: 'json' };
import LSP8MintableInit from '@lukso/lsp-smart-contracts/artifacts/LSP8MintableInit.json' assert { type: 'json'
};
import LSP9Vault from '@lukso/lsp-smart-contracts/artifacts/LSP9Vault.json' assert { type: 'json' };
import LSP9VaultInit from '@lukso/lsp-smart-contracts/artifacts/LSP9VaultInit.json' assert { type: 'json' };
import LSP16UniversalFactory from '@lukso/lsp-smart-contracts/artifacts/LSP16UniversalFactory.json'
assert { type: 'json' };
import LSP23LinkedContractsFactory from '@lukso/lsp-smart-
contracts/artifacts/LinkedContractsFactory.json' assert { type: 'json' };

export default {
    LSP0ERC725Account,
    LSP0ERC725AccountInit,
    LSP1UniversalReceiverDelegateUP,
    LSP1UniversalReceiverDelegateVault,
    LSP6KeyManager,
    LSP6KeyManagerInit,
    LSP7CappedSupply,
    LSP7CappedSupplyInitAbstract,
    LSP7Mintable,
    LSP7MintableInit,
    LSP8CappedSupply,
    LSP8CappedSupplyInitAbstract,
    LSP8Mintable,
    LSP8MintableInit,
    LSP9Vault,
    LSP9VaultInit,
    LSP16UniversalFactory,
    LSP23LinkedContractsFactory,
};
```

</file>

<file>

path: /src/select_rpc.ts

url: https://github.com/b00ste/up-debugging-cli/blob/main/src/select_rpc.ts

```
import inquirer from 'inquirer';

export const setRPC = async () => {
    const { networkName } = await inquirer.prompt({
        type: 'list',
        name: 'networkName',
        message: 'Choose network:',
        choices: ['LUKSO Mainnet', 'LUKSO Testnet', 'Goerli'],
        default: 'LUKSO Testnet',
    });
    if (networkName === 'LUKSO Mainnet') {
        const RPC = 'https://rpc.lukso.gateway.fm';
        const explorerBaseLink =
            'https://explorer.execution.mainnet.lukso.network/';

        global.RPC = RPC;

        return {
            networkName,
            explorerBaseLink,
            RPC,
        };
    } else if (networkName === 'LUKSO Testnet') {
        const RPC = 'https://rpc.testnet.lukso.gateway.fm';
        const explorerBaseLink =
            'https://explorer.execution.testnet.lukso.network/';

        global.RPC = RPC;

        return {
            networkName,
            explorerBaseLink,
            RPC,
        };
    } else if (networkName === 'Goerli') {
        const RPC = 'https://rpc.goerli.eth.gateway.fm';
        const explorerBaseLink = 'https://goerli.etherscan.io/';

        global.RPC = RPC;

        return { networkName, explorerBaseLink, RPC };
    } else throw new Error('Unexpected Error: No RPC was slected.');
};
```

</file>

<file>

path: /src/universal_profile_interactions/execute.interaction.ts

url: https://github.com/b00ste/up-debugging-cli/blob/main/src/universal_profile_interactions/execute.interaction.ts

```

```

</file>

<file>

path: /src/universal_profile_interactions/setData.interaction.ts

url: https://github.com/b00ste/up-debugging-cli/blob/main/src/universal_profile_interactions/setData.interaction.ts

```
import inquirer from 'inquirer';
import { Contract, hexlify, isHexString, toBeArray, toUtf8Bytes } from 'ethers';

import ERC725YDataKeys from '../../storage/ERC725YDataKeys.json' assert { type: 'json' };
import {
    OnlySpacesStringNotAllowed,
    isNumeric,
    verifyAddressValidity,
    verifyBytes32Validity,
    verifyBytes4Validity,
    verifyUint128Validity,
    verifyUint64Validity,
} from '../utils';

const askForDataValue = async () => {
    const { dataValue }: { dataValue: string } = await inquirer.prompt([
        {
            type: 'input',
            name: 'dataValue',
            message:
                'Please type the value that you want to set for the previously selected Data Key:\n',
        },
    ]);

    OnlySpacesStringNotAllowed(dataValue);

    if (!isHexString(dataValue)) {
        if (isNumeric(dataValue)) {
            return hexlify(toBeArray(dataValue));
        } else if (isNaN(dataValue as any)) {
            return toUtf8Bytes(dataValue);
        } else throw new Error( Error: Invalid Data Value. ('${dataValue}') );
    }

    return dataValue;
};
```

```
const dataValueValidityCheck = (
    standard: string,
    dataKeyName: string,
    rawDataValue: string,
) => {
    let dataValue: string;
    if (ERC725YDataKeys[standard][dataKeyName].dataValueType === 'address') {
        dataValue = verifyAddressValidity(rawDataValue);
    } else if (
        ERC725YDataKeys[standard][dataKeyName].dataValueType === 'bytes32'
    ) {
        dataValue = verifyBytes32Validity(rawDataValue);
    } else if (
        ERC725YDataKeys[standard][dataKeyName].dataValueType ===
        'bytes[CompactBytesArray]'
    ) {
    } else if (
        ERC725YDataKeys[standard][dataKeyName].dataValueType ===
        '(bytes4,address,bytes4,bytes4)[CompactBytesArray]'
    ) {
    } else if (
        ERC725YDataKeys[standard][dataKeyName].dataValueType ===
        '(bytes4,bytes8)'
    ) {
        const [bytes4, uint64] = rawDataValue.split(',');

        dataValue =
            verifyBytes4Validity(bytes4) +
            verifyUint64Validity(uint64).substring(2);
    }
    return dataValue;
};

const askForStandardRelatedDataKeyAndValue = async () => {
    // ---- Retrieve Data Key ----

    const LSPs = [];
    for (const LSP in ERC725YDataKeys) {
        LSPs.push(LSP);
    }

    const { standard }: { standard: string } = await inquirer.prompt({
        type: 'list',
        name: 'standard',
        message:
            'Please select a standard for which you want to change a Data Key:',
        choices: LSPs,
    });

    const dataKeys = [];
    for (const dataKey in ERC725YDataKeys[standard]) {
```

```
    dataKeys.push(dataKey);
}

const { dataKeyName }: { dataKeyName: string } = await inquirer.prompt({
    type: 'list',
    name: 'dataKeyName',
    message: 'Please select a Data Key that you want to change:',
    choices: dataKeys,
});

if (dataKeyName.endsWith('[]')) {
    const { lengthOrIndex }: { lengthOrIndex: string } =
        await inquirer.prompt({
            type: 'list',
            name: 'lengthOrIndex',
            message:
                'Do you want to modify the length of the array or an element at a specific index?',
            choices: ['length', 'index'],
        });

    if (lengthOrIndex === 'length') {
        const dataKey = ERC725YDataKeys[standard][dataKeyName].length;
        const { rawDataValue }: { rawDataValue: string } =
            await inquirer.prompt({
                type: 'input',
                name: 'rawDataValue',
                message:
                    'Please provide a new uint128 length for the array.\n',
            });

        const dataValue = verifyUint128Validity(rawDataValue);

        return { dataKey, dataValue };
    } else if (lengthOrIndex === 'index') {
        const { dataKeySuffix }: { dataKeySuffix: string } =
            await inquirer.prompt({
                type: 'input',
                name: 'dataKeySuffix',
                message:
                    'Please provide a uint128 index for the array element.\n',
            });

        const dataKey =
            ERC725YDataKeys[standard][dataKeyName].index +
            verifyUint128Validity(dataKeySuffix);

        const { rawDataValue }: { rawDataValue: string } =
            await inquirer.prompt({
                type: 'input',
                name: 'rawDataValue',
                message: 'Please provide an address for the element.\n',
            });
```

```
        const dataValue = verifyAddressValidity(rawDataValue);

        return { dataKey, dataValue };
    }
} else if (ERC725YDataKeys[standard][dataKeyName].dataKey.length === 66) {
    const dataKey = ERC725YDataKeys[standard][dataKeyName].dataKey;

    /// ----- RAW DATA VALUE RETRIEVAL ----- ///

    const { rawDataValue }: { rawDataValue: string } =
        await inquirer.prompt({
            type: 'input',
            name: 'rawDataValue',
            message:
                ERC725YDataKeys[standard][dataKeyName].dataValueMessage,
        });

    /// ----- DATA VALUE VALIDITY CHECK ----- ///

    const dataValue = dataValueValidityCheck(
        standard,
        dataKeyName,
        rawDataValue,
    );

    return { dataKey, dataValue };
} else if (ERC725YDataKeys[standard][dataKeyName].dataKey.length < 66) {
    /// ----- DATA KEY SUFIX RETRIEVAL ----- ///

    const dataKeyPrefix = ERC725YDataKeys[standard][dataKeyName].dataKey;
    const { dataKeySufix }: { dataKeySufix: string } =
        await inquirer.prompt({
            type: 'input',
            name: 'dataKeySufix',
            message: ERC725YDataKeys[standard][dataKeyName].dataKeyMessage,
        });

    /// ----- DATA KEY SUFIX VALIDITY CHECK ----- ///

    let dataKey: string;
    if (ERC725YDataKeys[standard][dataKeyName].dataKeyType === 'address') {
        verifyAddressValidity(dataKeySufix);
        dataKey = dataKeyPrefix + dataKeySufix.substring(2);
    } else if (
        ERC725YDataKeys[standard][dataKeyName].dataKeyType === 'bytes4'
    ) {
        verifyBytes4Validity(dataKeySufix);
        dataKey =
            dataKeyPrefix + dataKeySufix.substring(2) + '0'.repeat(32);
    } else if (
        ERC725YDataKeys[standard][dataKeyName].dataKeyType === 'bytes32'
```

```
        ) {
            verifyBytes32Validity(dataKeySufix);
            dataKey = dataKeyPrefix + dataKeySufix.substring(2, 42);
        }

        /// ----- RAW DATA VALUE RETRIEVAL ----- ///

        const { rawDataValue }: { rawDataValue: string } =
            await inquirer.prompt({
                type: 'input',
                name: 'rawDataValue',
                message:
                    ERC725YDataKeys[standard][dataKeyName].dataValueMessage,
            });

        /// ----- DATA VALUE VALIDITY CHECK ----- ///

        const dataValue = dataValueValidityCheck(
            standard,
            dataKeyName,
            rawDataValue,
        );

        return { dataKey, dataValue };
    } else throw new Error('Error: Invalid standardized data key');
};

const askForArbitraryRelatedDataKeyAndValue = async () => {
    // ---- Retrieve Data Key ----

    const { dataKey }: { dataKey: string } = await inquirer.prompt([
        {
            type: 'input',
            name: 'dataKey',
            message:
                'Please type a Data Key that you want to update the value for:\n',
        },
    ]);

    verifyBytes32Validity(dataKey);

    // ---- Retrieve Data Value ----

    const dataValue = await askForDataValue();

    return { dataKey, dataValue };
};

export const setData = async (UniversalProfile: Contract) => {
    const { updateLSPSpecificDataKey } = await inquirer.prompt({
        type: 'list',
        name: 'updateLSPSpecificDataKey',
```

```
        message: 'Do you want to update Data Key related to the LSPs?',
        choices: ['yes', 'no'],
        default: 'yes',
    });

    if (updateLSPSpecificDataKey === 'yes') {
        const { dataKey, dataValue } =
            await askForStandardRelatedDataKeyAndValue();

        return await UniversalProfile.setData(dataKey, dataValue);
    } else if (updateLSPSpecificDataKey === 'no') {
        const { dataKey, dataValue } =
            await askForArbitraryRelatedDataKeyAndValue();

        return await UniversalProfile.setData(dataKey, dataValue);
    } else throw new Error("Unexpected Error: Answer must be 'yes' or 'no'.");
};

export const setDataBatch = async (UniversalProfile: Contract) => {
    const { numberOfDataKeys }: { numberOfDataKeys: string } =
        await inquirer.prompt({
            type: 'input',
            name: 'numberOfDataKeys',
            message: '',
        });

    if (!isNumeric(numberOfDataKeys)) throw new Error('Error: Not a number.');

    const dataKeys = [];
    const dataValues = [];
    for (let i = 0; i < Number.parseInt(numberOfDataKeys); i++) {
        const { updateLSPSpecificDataKey } = await inquirer.prompt({
            type: 'list',
            name: 'updateLSPSpecificDataKey',
            message: 'Do you want to update Data Key related to the LSPs?',
            choices: ['yes', 'no'],
            default: 'yes',
        });

        if (updateLSPSpecificDataKey === 'yes') {
            const { dataKey, dataValue } =
                await askForStandardRelatedDataKeyAndValue();

            dataKeys.push(dataKey);
            dataValues.push(dataValue);
        } else if (updateLSPSpecificDataKey === 'no') {
            const { dataKey, dataValue } =
                await askForArbitraryRelatedDataKeyAndValue();

            dataKeys.push(dataKey);
            dataValues.push(dataValue);
        } else
```

```
            throw new Error("Unexpected Error: Answer must be 'yes' or 'no'.");
    }
    return await UniversalProfile.setDataBatch(dataKeys, dataValues);
};
```

</file>

<file>

path: /src/utils.ts

url: https://github.com/b00ste/up-debugging-cli/blob/main/src/utils.ts

```
import {
    hexlify,
    isHexString,
    keccak256,
    toBeArray,
    toUtf8Bytes,
} from 'ethers';

export const isNumeric = (value: string | number): boolean => {
    return value != null && value !== '' && !isNaN(Number(value.toString()));
};

export const OnlySpacesStringNotAllowed = (text: string) => {
    if (text.replaceAll(' ', '').length === 0)
        throw new Error( Error: Salt not valid. ('${text}') );
};

export const verifyAddressValidity = (address: string) => {
    return verifyBytesNValidity(address, 20, false, false, false, true);
};
export const verifyBytes4Validity = (bytes4: string) => {
    return verifyBytesNValidity(bytes4, 4, false, false, true, true);
};
export const verifyBytes32Validity = (bytes32: string) => {
    return verifyBytesNValidity(bytes32, 20, false, false, true, true);
};
export const verifySaltValidity = (salt: string) => {
    return verifyBytesNValidity(salt, 20, true, true, true, true);
};
export const verifyUint64Validity = (uint64: string) => {
    return verifyBytesNValidity(uint64, 8, true, false, false, true);
};
export const verifyUint128Validity = (uint128: string) => {
    return verifyBytesNValidity(uint128, 16, true, false, false, true);
};

export const verifyBytesNValidity = (
    bytesNValue: string,
```

```
  N: number,
  lessBytesAllowed?: boolean,
  moreBytesAllowed?: boolean,
  textAllowed?: boolean,
  numberAllowed?: boolean,
) => {
  OnlySpacesStringNotAllowed(bytesNValue);

  const requiredNumberOfCharacters = 2 + N * 2;

  if (isHexString(bytesNValue)) {
    if (bytesNValue.length === requiredNumberOfCharacters) {
      return bytesNValue;
    } else if (
      lessBytesAllowed &&
      bytesNValue.length < requiredNumberOfCharacters
    ) {
      const missingZeros =
        requiredNumberOfCharacters - bytesNValue.length;
      const paddedHexValue =  0x${
        '0'.repeat(missingZeros) + bytesNValue.substring(2)
      } ;
      return paddedHexValue;
    } else if (
      moreBytesAllowed &&
      bytesNValue.length > requiredNumberOfCharacters
    ) {
      return keccak256(bytesNValue).substring(
        0,
        requiredNumberOfCharacters,
      );
    } else
      throw new Error(
         Error: Invalid bytes${N}. Length must be ${N} bytes. ,
      );
  } else {
    if (numberAllowed && isNumeric(bytesNValue)) {
      const hexValue = hexlify(toBeArray(bytesNValue));
      if (hexValue.length > requiredNumberOfCharacters)
        throw new Error(
          'Error: Overflow. Number too big for storage type.',
        );
      const missingZeros = requiredNumberOfCharacters - hexValue.length;
      const paddedHexValue =  0x${
        '0'.repeat(missingZeros) + hexValue.substring(2)
      } ;
      return paddedHexValue;
    } else if (textAllowed && isNaN(bytesNValue as any)) {
      return keccak256(toUtf8Bytes(bytesNValue));
    } else
      throw new Error(
         Error: Invalid bytes${N} value. ('${bytesNValue}') ,
```

```
            );
    }
};

export function getProxyBytecode(baseContractAddress: string): string {
    /**
     * @see https://blog.openzeppelin.com/deep-dive-into-the-minimal-proxy-contract/
     * The first 10 x hex opcodes copy the runtime code into memory and return it.
     */
    const eip1167RuntimeCodeTemplate =

'0x3d602d80600a3d3981f3363d3d373d3d3d363d73bebebebebebebebebebebebebebebebebebebebe5af43
d82803e903d91602b57fd5bf3';

    // deploy proxy contract
    const proxyBytecode = eip1167RuntimeCodeTemplate.replace(
        'bebebebebebebebebebebebebebebebebebebebe',
        baseContractAddress.substr(2),
    );

    return proxyBytecode;
}
```

</file>

<file>

path: /storage/ERC725YDataKeys.json

url: https://github.com/b00ste/up-debugging-cli/blob/main/storage/ERC725YDataKeys.json

```
{
    "LSP1": {
        "LSP1UniversalReceiverDelegatePrefix": {
            "dataKey": "0x0cfc51aec37c55a4d0b10000",
            "dataKeyMessage": "Please provide a typeId that is handled in the Universal Receiver Delegate that
you want to add.\n(must be a valid bytes32, a string or a number)\n",
            "dataKeyType": "bytes32",
            "dataValueMessage": "Please provide the address of the Universal Receiver Delegate that you want
to add.\n(must be a valid etehreum address or a number)\n",
            "dataValueType": "address"
        },
        "LSP1UniversalReceiverDelegate": {
            "dataKey": "0x0cfc51aec37c55a4d0b1a65c6255c4bf2fbdf6277f3cc0730c45b828b6db8b47",
            "dataValueMessage": "Please provide the address of the Universal Receiver Delegate that you want
to add.\n(must be a valid etehreum address or a number)\n",
            "dataValueType": "address"
        }
    },
    "LSP3": {
        "SupportedStandards_LSP3": {
```

```
        "key": "0xeafec4d89fa9619884b60000abe425d64acd861a49b8ddf5c0b6962110481f38",
        "value": "0xabe425d6"
      },

      "LSP3Profile": {
        "dataKey": "0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5",
        "dataValueMessage": "Please provide the metadata for your Universal Profile.",
        "dataValueType": "bytes"
      }
    },
    "LSP6": {
      "AddressPermissions[]": {
        "length": "0xdf30dba06db6a30e65354d9a64c609861f089545ca58c6b4dbe31a5f338cb0e3",
        "index": "0xdf30dba06db6a30e65354d9a64c60986"
      },

      "AddressPermissions:Permissions": {
        "dataKey": "0x4b80742de2bf82acb3630000",
        "dataKeyMessage": "Please provide a controller address whose Permissions you want to
update.\n(must be a valid etehreum address or a number)\n",
        "dataKeyType": "address",
        "dataValueMessage": "Please provide a Permissions BitArray for the controller you want to
set.\n(must be a valid bytes32, a string or a number)\n",
        "dataValueType": "bytes32"
      },

      "AddressPermissions:AllowedERC725YDataKeys": {
        "dataKey": "0x4b80742de2bf866c29110000",
        "dataKeyMessage": "Please provide a controller address whose AllowedERC725YDataKeys you
want to update.\n(must be a valid etehreum address or a number)\n",
        "dataKeyType": "address",
        "dataValueMessage": "Please provide a CompactBytesArray of AllowedERC725YDataKeys for the
controller you want to set.\n(must be a valid CompactBytesArray of bytes)\n",
        "dataValueType": "bytes[CompactBytesArray]"
      },

      "AddressPermissions:AllowedCalls": {
        "dataKey": "0x4b80742de2bf393a64c70000",
        "dataKeyMessage": "Please provide a controller address whose AllowedCalls you want to
update.\n(must be a valid etehreum address or a number)\n",
        "dataKeyType": "address",
        "dataValueMessage": "Please provide a CompactBytesArray of AllowedCalls for the controller you
want to set.\n(must be a valid CompactBytesArray of (bytes4,address,bytes4,bytes4))\n",
        "dataValueType": "(bytes4,address,bytes4,bytes4)[CompactBytesArray]"
      }
    },
    "LSP12": {
      "LSP12IssuedAssetsMap": {
        "dataKey": "0x74ac2555c10b9349e78f0000",
        "dataKeyMessage": "Please provide an address of the asset that you have issued.\n(must be a valid
etehreum address or a number)\n",
        "dataKeyType": "address",
```

```
            "dataValueMessage": "Please provide the bytes4 interface id and the bytes8 index belonging to the issued asset. (e.g. 0xcafecafe, 15)",
            "dataValueType": "(bytes4,bytes8)"
        },

        "LSP12IssuedAssets[]": {
            "length": "0x7c8c3416d6cda87cd42c71ea1843df28ac4850354f988d55ee2eaa47b6dc05cd",
            "index": "0x7c8c3416d6cda87cd42c71ea1843df28"
        }
    },
    "LSP17": {
        "LSP17ExtensionPrefix": {
            "dataKey": "0xcee78b4094da860110960000",
            "dataKeyMessage": "Please provide a bytes4 selector of a method that you want to add as an extension.\n(must be a valid bytes4, a string or a number)\n",
            "dataKeyType": "bytes4",
            "dataValueMessage": "Please provide the address of the extension contract that you want to set.\n(must be a valid etehreum address or a number)\n",
            "dataValueType": "address"
        }
    }
}
```


</file>

<file>

path: /storage/PublicSecrets.json

url: https://github.com/b00ste/up-debugging-cli/blob/main/storage/PublicSecrets.json


```
{
    "aaa": "0xe86B13ee8fa8FBc3FaF4A45bA7aa4BDD694B4A6A",
    "abc": "0xE402a27cD3E8901b6770778aCc67D6e5c6A1004A"
}
```


</file>

<file>

path: /storage/UsedSalts.json

url: https://github.com/b00ste/up-debugging-cli/blob/main/storage/UsedSalts.json

```
{
   "5": {
      "lsp16": {
         "LSP0ERC725Account": {
            "0xE402a27cD3E8901b6770778aCc67D6e5c6A1004A": {
               "0xc89efdaa54c0f20c7adf612882df0950f5a951637e0307cdcb4c672f298b8bc6":
"0xE3Abf598211E7ebc0AEd7ffcd53C359D05c0ACC4",
               "0xad7c5bef027816a800da1736444fb58a807ef4c9603b7848673f7e3a68eb14a5":
"0x53F721B68cD86ab954a264296161b10CcD0fE78b"
            },
            "0xe86B13ee8fa8FBc3FaF4A45bA7aa4BDD694B4A6A": {
               "0xc89efdaa54c0f20c7adf612882df0950f5a951637e0307cdcb4c672f298b8bc6":
"0xC57BAeC8a29FB626D926B10b022485530ab30aeA",
               "0xad7c5bef027816a800da1736444fb58a807ef4c9603b7848673f7e3a68eb14a5":
"0xA74245947c3CB0e4c5C07FF646a68861A97011Eb"
            }
         }
      },
      "lsp23": {}
   },
   "4201": {
      "lsp16": {
         "LSP0ERC725Account": {
            "0xe86B13ee8fa8FBc3FaF4A45bA7aa4BDD694B4A6A": {
               "0x0000000000000000000000000000000000000000000000000000000000000001":
"0x8EeA12bdCeAb06314076AccA0acF0a6D7b65e309",
               "0x0000000000000000000000000000000000000000000000000000000000000002":
"0x5b64608ab1c0338f65d1F49cAe63B1F9dCD06b92"
            },
            "0xE402a27cD3E8901b6770778aCc67D6e5c6A1004A": {
               "0x0000000000000000000000000000000000000000000000000000000000000001":
"0xa8032Fa7c6B9C4A37EF0Af6E6898b71d4ef528A1",
               "0x0000000000000000000000000000000000000000000000000000000000000002":
"0x78f49A5Fd94CDBb55A71c45b0A74a8250714DB15",
               "0x0000000000000000000000000000000000000000000000000000000000000003":
"0xE5D4876B8877601407c57488Ed85F949269E971B",
               "0x0000000000000000000000000000000000000000000000000000000000000004":
"0xF46d8863DD8fD09405f37369Ae471c9Bf2BecDd8"
            },
            "0x00000000000000000000000e86B13ee8fa8FBc3FaF4A45bA7aa4BDD694B4A6A": {
               "0x0000ca263f54ffE8E7419f134aF27A27cb94De2C":
"0xef15a5bb5a98bf865fa1279974cfb703bc3abdcf0f91b1c88a8c95c218e12083"
            }
         }
      },
      "lsp23": {}
   }
}
```


</file>

<file>

path: /tsconfig.json

url: https://github.com/b00ste/up-debugging-cli/blob/main/tsconfig.json

```
{
    "compilerOptions": {
        "target": "ES6",
        "module": "ESNext",
        "moduleResolution": "Node",
        "esModuleInterop": true,
        "resolveJsonModule": true,
        "outDir": "dist",
        "forceConsistentCasingInFileNames": true,
        "skipLibCheck": true,
        "lib": ["ES2021.String"]
    },
    "include": ["./src", "./index.ts"],
    "exclude": ["/node_modules"],
    "ts-node": {
        "esm": true,
        "experimentalSpecifierResolution": "node"
    },
    "emitDecoratorMetadata": true,
    "experimentalDecorators": true
}
```

</file>

</repository>

<repository>

tools-eip191-signer

overview

repository: https://github.com/lukso-network/tools-eip191-signer/
userName: lukso-network
repository: tools-eip191-signer
branch: main
date: 2023-11-27T10:50:57+01:00 (1701078657)

<file>

path: /.all-contributorsrc

url: https://github.com/lukso-network/tools-eip191-signer/blob/main/.all-contributorsrc


```
{
  "projectName": "tools-eip191-signer",
  "projectOwner": "lukso-network",
  "repoType": "github",
  "repoHost": "https://github.com",
  "files": [
    "README.md"
  ],
  "imageSize": 50,
  "commit": false,
  "commitConvention": "eslint",
  "contributors": [
    {
      "login": "magalimorin18",
      "name": "Magali Morin",
      "avatar_url": "https://avatars.githubusercontent.com/u/51906903?v=4",
      "profile": "https://github.com/magalimorin18",
      "contributions": [
        "code",
        "test"
      ]
    },
    {
      "login": "frozeman",
      "name": "Fabian Vogelsteller",
      "avatar_url": "https://avatars.githubusercontent.com/u/232662?v=4",
      "profile": "https://lukso.network/",
      "contributions": [
        "ideas"
      ]
    },
    {
```

```
    "login": "CallumGrindle",
    "name": "Callum Grindle",
    "avatar_url": "https://avatars.githubusercontent.com/u/54543428?v=4",
    "profile": "https://github.com/CallumGrindle",
    "contributions": [
      "review",
      "mentoring"
    ]
  },
  {
    "login": "Hugoo",
    "name": "Hugo Masclet",
    "avatar_url": "https://avatars.githubusercontent.com/u/477945?v=4",
    "profile": "http://www.hugomasclet.com/",
    "contributions": [
      "review",
      "mentoring"
    ]
  }
 ],
 "contributorsPerLine": 7
}
```


</file>

<file>

path: /.eslintrc.json

url: https://github.com/lukso-network/tools-eip191-signer/blob/main/.eslintrc.json

```
{
  "env": {
    "browser": true,
    "es2021": true,
    "jest": true
  },
  "extends": [
    "eslint:recommended",
    "prettier",
    "plugin:@typescript-eslint/recommended",
    "plugin:prettier/recommended",
    "plugin:import/recommended",
    "plugin:import/typescript"
  ],
  "parser": "@typescript-eslint/parser",
  "parserOptions": {
    "ecmaVersion": "latest"
  },
  "plugins": ["@typescript-eslint", "prettier", "jest", "import"],
  "rules": {
    "prettier/prettier": "error",
    "import/order": [
      "error",
      {
        "groups": [["external", "internal"]],
        "newlines-between": "always",
        "alphabetize": { "order": "asc" }
      }
    ]
  }
}
```

</file>

<file>

path: /.npmignore

url: https://github.com/lukso-network/tools-eip191-signer/blob/main/.npmignore

```
/src
/test
docs
.github
.prettierrc
.eslintrc
/node_modules
```

</file>

<file>

path: /.prettierrc

url: https://github.com/lukso-network/tools-eip191-signer/blob/main/.prettierrc

```
{
  "trailingComma": "all",
  "tabWidth": 2,
  "semi": true,
  "singleQuote": true,
  "printWidth": 80
}
```

</file>

<file>

path: /CHANGELOG.md

url: https://github.com/lukso-network/tools-eip191-signer/blob/main/CHANGELOG.md

Changelog

0.2.2 (https://github.com/lukso-network/tools-eip191-signer/compare/v0.2.1...v0.2.2) (2023-01-31)

Documentation

- update links in docs (80eac41 (https://github.com/lukso-network/tools-eip191-signer/commit/80eac4160b0e26dd868b8edb302b7655c265953b))

0.2.1 (https://github.com/lukso-network/tools-eip191-signer/compare/v0.2.0...v0.2.1) (2022-10-28)

Features

- add eslint import rules (36f6981 (https://github.com/lukso-network/tools-eip191-signer/commit/36f698192a9abfba398fe4defdfe66ef0d2cc884))

Bug Fixes

- update README (f26910e (https://github.com/lukso-network/tools-eip191-signer/commit/f26910e2015d188dd22a28c942a6692fbda1b71c))

0.2.0 (https://github.com/lukso-network/tools-eip191-signer/compare/v0.1.0...v0.2.0) (2022-10-26)

Bug Fixes

- improve test (83a0781 (https://github.com/lukso-network/tools-eip191-signer/commit/83a0781a9f2e3bc454f15525d4ea63abc26c19d8))
- refactor code and add test (e2b6fdb (https://github.com/lukso-network/tools-eip191-signer/commit/e2b6fdb57f7eb08d944dea1540199598e7356a8c))
- update docs and variable name for clarity (64ff1ce (https://github.com/lukso-network/tools-eip191-signer/commit/64ff1ce1dcc6e6065d25540241bfc4abf3d51df6))

Miscellaneous Chores

- release 0.2.0 (6697005 (https://github.com/lukso-network/tools-eip191-signer/commit/6697005d10f40ccf341898d3f05839f6b1899151))

Breaking Changes

- change package name (cb2c3ef (https://github.com/lukso-network/tools-eip191-signer/commit/cb2c3efe95cd6728d76fefd3090720fb8f78b1b6))

0.1.0 (https://github.com/lukso-network/tools-eip191-signer/compare/v0.0.2-development...v0.1.0) (2022-10-07)

Bug Fixes

- adapt code to prefix '\x19Execute Relay Call:\n' (dcb8674 (https://github.com/lukso-network/tools-eip191-signer/commit/dcb86741974f3437b13ae2fdd95d13118ca290be))

Miscellaneous Chores

- release 0.1.0 (2bd42a6 (https://github.com/lukso-network/tools-eip191-signer/commit/2bd42a6ec50f978c5855b7fb9a4e46440d99466d))

0.0.2-development (https://github.com/lukso-network/tools-eip191-signer/compare/v0.0.1-development...v0.0.2-development) (2022-09-30)

Features

- add package to specify contributors (33404a4 (https://github.com/lukso-network/tools-eip191-signer/commit/33404a4348410d550c6324f847ab7d4da05fc353))

Bug Fixes

- change the branch to open PR against to main (25c9f60 (https://github.com/lukso-network/tools-eip191-signer/commit/25c9f60535da22d5065c0ee5a8a9b0c3541d7682))
- change the default branch for the release workflow to main (d99b7d5 (https://github.com/lukso-network/tools-eip191-signer/commit/d99b7d5383629ec75634222ac4ec09a701a57b8c))

0.0.1-development (2022-09-26)

Bug Fixes

- add checkout step to the release workflow (ad0d037 (https://github.com/lukso-network/tools-eip191-signer/commit/ad0d0372ee306279ceb6a8a21e3c120d3163a29d))
- set bump-patch-for-minor-pre-major for the release (9cd1f67 (https://github.com/lukso-network/tools-eip191-signer/commit/9cd1f67eb55fef1cb98a8444ade3379c5fe4d2c7))

Miscellaneous Chores

- release 0.0.1-development (5305ad1 (https://github.com/lukso-network/tools-eip191-signer/commit/5305ad1c9cd8569a12852759c51709b60c848fda))
  </file>

<file>

path: /CONTRIBUTING.md

url: https://github.com/lukso-network/tools-eip191-signer/blob/main/CONTRIBUTING.md

Contributing to eip191-signer.js

Commits

You should use Conventional Commit messages (https://www.conventionalcommits.org/).

The Conventional Commits specification is a lightweight convention on top of commit messages.
It provides an easy set of rules for creating an explicit commit history;
which makes it easier to write automated tools on top of.

The most important prefixes you should have in mind are:

- fix: which represents bug fixes, and correlates to a SemVer (https://semver.org/) patch.
- feat: which represents a new feature, and correlates to a SemVer minor.
- feat!:, or fix!:, refactor!:, etc., which represent a breaking change (indicated by the !) and will result in a SemVer major.

Other prefixes are also allowed :

- build: Changes that affect the build system or external dependencies.
- ci: Changes to our CI configuration files and scripts.
- docs: Documentation only changes.
- perf: A code change that improves performance.
- refactor: A code change that neither fixes a bug nor adds a feature.
- style: Changes that do not affect the meaning of the code (white-space, formatting, missing semi-colons, etc).
- test: Adding missing tests or correcting existing tests.
- chore: Other

Further details on conventional commits can be found here: https://www.conventionalcommits.org/en/v1.0.0/ (https://www.conventionalcommits.org/en/v1.0.0/)

Building

```
npm run build
```

This will build the library into /build.

Testing

```
yarn test
```

Become a contributor

To become a contributor please follow the [all-contributors (https://github.com/all-contributors/all-contributors)](https://github.com/all-contributors/all-contributors) specification.
</file>

<file>

path: /README.md

url: https://github.com/lukso-network/tools-eip191-signer/blob/main/README.md

eip191-signer.js ·  [(./LICENSE)](./LICENSE)  [(https://www.npmjs.com/package/@lukso/eip191-signer.js)](https://www.npmjs.com/package/@lukso/eip191-signer.js)  [(https://github.com/lukso-network/tools-eip191-signer/pulls)](https://github.com/lukso-network/tools-eip191-signer/pulls)

<h2 align="center">@lukso/eip191-signer.js</h2>
Helper library to sign any EIP191 data.

For more information see Documentation.

Getting Started

The @lukso/eip191-signer.js package is used to sign messages according to the [EIP191 standard (https://eips.ethereum.org/EIPS/eip-191)](https://eips.ethereum.org/EIPS/eip-191).

If you want to sign with the version 0x45, then use the function signEthereumSignedMessage.

If you want to sign with the version 0x00, then use the function signDataWithIntendedValidator.

- [GitHub Repository (https://github.com/lukso-network/tools-eip191-signer)](https://github.com/lukso-network/tools-eip191-signer)
- [NPM Package (https://www.npmjs.com/package/@lukso/eip191-signer.js)](https://www.npmjs.com/package/@lukso/eip191-signer.js)

Install

```
npm install @lukso/eip191-signer.js
```

Usage

```
import { EIP191Signer } from '@lukso/eip191-signer.js';

const eip191Signer = new EIP191Signer();
```

hashEthereumSignedMessage

```
eip191Signer.hashEthereumSignedMessage(message);
```

Hashes the given message with the version 0x45.

The message will be enveloped as follows: '\x19' + '\x45' + 'thereum Signed Message:\n' + messageBytes.length + message and hashed using keccak256.

hashDataWithIntendedValidator

```
eip191Signer.hashDataWithIntendedValidator(validatorAddress, message);
```

Hashes the given message with the version 0x00.

The message will be enveloped as follows: '\x19' + '\x00' + validatorAddress + message and hashed using keccak256.

signEthereumSignedMessage

```
eip191Signer.signEthereumSignedMessage(message, signingKey);
```

This method is for signing a message with the version 0x45.

The message passed as parameter will be wrapped as follows: '\x19' + '\x45' + 'thereum Signed Message:\n' + messageBytes.length + message.

signDataWithIntendedValidator

```
eip191Signer.signDataWithIntendedValidator(
  validatorAddress,
  message,
  signingKey,
);
```

This method is for signing a message with the version 0x00.

The message passed as parameter will be wrapped as follows: '\x19' + '\x00' + validatorAddress + message.

recover

```
eip191Signer.recover(messageHash, signature);
```

Recovers the address which was used to sign the given message.

Contributing

Please check CONTRIBUTING.md (./CONTRIBUTING.md).

License

eip191-signer.js is Apache 2.0 licensed (./LICENSE).

Contributors

<!-- ALL-CONTRIBUTORS-LIST:START - Do not remove or modify this section -->
<!-- prettier-ignore-start -->
<!-- markdownlint-disable -->
<table>
<tbody>
<tr>
<td align="center"><img
src="https://avatars.githubusercontent.com/u/51906903?v=4?s=50" width="50px;" alt="Magali Morin"/>
_{Magali Morin}
  ⚠</td>
<td align="center"><img
src="https://avatars.githubusercontent.com/u/232662?v=4?s=50" width="50px;" alt="Fabian Vogelsteller"/>

_{Fabian Vogelsteller}
 </td>
<td align="center"><img
src="https://avatars.githubusercontent.com/u/54543428?v=4?s=50" width="50px;" alt="Callum Grindle"/>
_{Callum Grindle}
   </td>
<td align="center"><img
src="https://avatars.githubusercontent.com/u/477945?v=4?s=50" width="50px;" alt="Hugo Masclet"/>
_{Hugo Masclet}
   </td>
</tr>
</tbody>
</table>

<!-- markdownlint-restore -->
<!-- prettier-ignore-end -->

<!-- ALL-CONTRIBUTORS-LIST:END -->
</file>

<file>

path: /RELEASE.md

url: https://github.com/lukso-network/tools-eip191-signer/blob/main/RELEASE.md

Release Process

Releases are published to npm using Release Please (https://github.com/googleapis/release-please).

This package automates CHANGELOG generation, version bumps and npm release by parsing the git history, looking for Conventional Commit messages (https://www.conventionalcommits.org/), and creating release PRs from develop into main. Release PRs are kept up-to-date as additional work is merged into main.

In order to tag a release, simply merge the release PR from develop into main. Both squash-merge and merge commits work with Release PRs.

After having merged the release PR into main, release-please takes the following steps:

1. Updates the changelog file (for example CHANGELOG.md), along with other language specific files (for example package.json).
2. Tags the commit with the version number.
3. Creates a GitHub Release based on the tag.
4. Release the package to npm.

The status label on the PR indicates where the release PR is in its lifecycle :

- autorelease: pending is the initial state of the Release PR before it is merged
- autorelease: tagged means that the Release PR has been merged and the release has been tagged in GitHub
- autorelease: snapshot is a special state for snapshot version bumps
- autorelease: published means that an npm release have been published based on the release PR.

How should I write my commits?

You should use Conventional Commit messages (https://www.conventionalcommits.org/).

The most important prefixes you should have in mind are:

- fix: which represents bug fixes, and correlates to a SemVer (https://semver.org/) patch.
- feat: which represents a new feature, and correlates to a SemVer minor.
- feat!:, or fix!:, refactor!:, etc., which represent a breaking change (indicated by the !) and will result in a SemVer major.

Other prefixes are also allowed :

- build: Changes that affect the build system or external dependencies.
- ci: Changes to our CI configuration files and scripts.
- docs: Documentation only changes.
- perf: A code change that improves performance.
- refactor: A code change that neither fixes a bug nor adds a feature.
- style: Changes that do not affect the meaning of the code (white-space, formatting, missing semi-colons, etc).
- test: Adding missing tests or correcting existing tests.
- chore: Other

What if my PR contains multiple fixes or features?

You can represent multiple changes in a single commit,
using footers:

```
feat: adds v4 UUID to crypto

This adds support for v4 UUIDs to the library.

fix(utils): unicode no longer throws exception
  PiperOrigin-RevId: 345559154
  BREAKING-CHANGE: encode method no longer throws.
  Source-Link: googleapis/googleapis@5e0dcb2

feat(utils): update encode to support unicode
  PiperOrigin-RevId: 345559182
  Source-Link: googleapis/googleapis@e5eef86
```

The above commit message will contain:

1. an entry for the "adds v4 UUID to crypto" feature.
2. an entry for the fix "unicode no longer throws exception", along with a note
   that it's a breaking change.
3. an entry for the feature "update encode to support unicode".

> :warning: Important: The additional messages must be added to the bottom of the commit.

How do I change the version number?

When a commit to the main branch has Release-As: x.x.x (case insensitive) in the commit body, Release Please will open a new pull request for the specified version.

Empty commit example:

git commit --allow-empty -m "chore: release 2.0.0" -m "Release-As: 2.0.0" results in the following commit message:

```
chore: release 2.0.0

Release-As: 2.0.0
```

How can I fix release notes?

If you have merged a pull request and would like to amend the commit message
used to generate the release notes for that commit, you can edit the body of
the merged pull requests and add a section like:

```
BEGIN_COMMIT_OVERRIDE
feat: add ability to override merged commit message

fix: another message
chore: a third message
END_COMMIT_OVERRIDE
```

The next time Release Please runs, it will use that override section as the
commit message instead of the merged commit message.
</file>

<file>

path: /docs/Classes/EIP191Signer.md

url: https://github.com/lukso-network/tools-eip191-signer/blob/main/docs/Classes/EIP191Signer.md

---

sidebar_position: 1

EIP191Signer

hashEthereumSignedMessage

```
eip191Signer.hashEthereumSignedMessage(message);
```

Hashes the given message. The message will be enveloped as follows: '\x19' + '\x45' + 'thereum Signed Message:\n' + messageBytes.length + message and hashed using keccak256.

Parameters

message - String: A message to hash.

Returns

String: The hashed message constructed as '\x19' + '\x45' + 'thereum Signed Message:\n' + messageBytes.length + message

Example

```
eip191Signer.hashEthereumSignedMessage('Hello World');
// '0xa1de988600a42c4b4ab089b619297c17d53cffae5d5120d82d8a92d0bb3b78f2';
```

hashDataWithIntendedValidator

```
eip191Signer.hashDataWithIntendedValidator(validatorAddress, message);
```

Hashes the given message. The message will be enveloped as follows: '\x19' + '\x00' + validatorAddress + message and hashed using keccak256.

Parameters

validatorAddress - String: The address of the validator.

message - String: A message to hash.

Returns

String: The hashed message constructed as '\x19' + '\x00' + validatorAddress + message

Example

```
eip191Signer.hashDataWithIntendedValidator(
  0xad278a6ead89f6b6c6fdf54a3e6e876660593b45,
  'Hello World',
);
// '0xa63022286ecaa3317625e319a64b3bf01c41da558dfc1890e8cb196eb414ffd5';
```

signEthereumSignedMessage

```
eip191Signer.signEthereumSignedMessage(message, signingKey);
```

Signs a message. The message passed as parameter will be wrapped as follows: '\x19' + '\x45' + 'thereum Signed Message:\n' + messageBytes.length + message

Parameters

1. message - String: The message to sign.
2. signingKey - String: The private key to sign with.

Returns

Object: The Message object

- message - String: The given message.
- messageHash - String: The hash of the given message constructed as '\x19' + '\x45' + 'thereum Signed Message:\n' + messageBytes.length + message.
- r - String: First 32 bytes of the signature.
- s - String: Next 32 bytes of the signature.
- v - String: Recovery value + 27.
- signature - String: The raw RLP encoded signature.

Example

```
eip191Signer.signEthereumSignedMessage(
  'Hello World',
  'ffeb17b9a6059fec3bbab63d76b060b7380cac7a62ce6621a134531a46458968',
);
/**
{    message: 'Hello World',
     messageHash: '0xa1de988600a42c4b4ab089b619297c17d53cffae5d5120d82d8a92d0bb3b78f2',
     v: '0x1c',
     r: '0x85c15865f2909897c1be6d66c1d9c86d6125978aec9e28d1a69d4d306bde694f',
     s: '0x7cf9723f0eeaf8815e3fa984ac1d7bf3c420786ead91abd4dd9c1657897efec1',
     signature:
'0x85c15865f2909897c1be6d66c1d9c86d6125978aec9e28d1a69d4d306bde694f7cf9723f0eeaf8815e3fa98
4ac1d7bf3c420786ead91abd4dd9c1657897efec11c'
   }
*/
```

signDataWithIntendedValidator

```
eip191Signer.signDataWithIntendedValidator(
  validatorAddress,
  message,
  signingKey,
);
```

Signs a message. The message passed as parameter will be wrapped as follows: '\x19' + '\x00' + validatorAddress + message

Parameters

1. validatorAddress - String: The address of the validator.
2. message - String: The message to sign.
3. signingKey - String: The private key to sign with.

Returns

Object: The Message object

- message - String: The given message.
- messageHash - String: The hash of the given message constructed as '\x19' + '\x00' + validatorAddress + message.
- r - String: First 32 bytes of the signature.
- s - String: Next 32 bytes of the signature.
- v - String: Recovery value + 27.
- signature - String: The raw RLP encoded signature.

Example

```
eip191Signer.signDataWithIntendedValidator(
  '0xad278a6ead89f6b6c6fdf54a3e6e876660593b45',
  'Hello World',
  'ffeb17b9a6059fec3bbab63d76b060b7380cac7a62ce6621a134531a46458968',
);
/**
   {
     message: 'Hello World',
     messageHash: '0xa1de988600a42c4b4ab089b619297c17d53cffae5d5120d82d8a92d0bb3b78f2',
     v: '0x1c',
     r: '0x85c15865f2909897c1be6d66c1d9c86d6125978aec9e28d1a69d4d306bde694f',
     s: '0x7cf9723f0eeaf8815e3fa984ac1d7bf3c420786ead91abd4dd9c1657897efec1',
     signature:
'0x85c15865f2909897c1be6d66c1d9c86d6125978aec9e28d1a69d4d306bde694f7cf9723f0eeaf8815e3fa98
4ac1d7bf3c420786ead91abd4dd9c1657897efec11c'
   }

*/
```

recover

```
eip191Signer.recover(message, signature);
```

Recovers the address which was used to sign the given message.

Parameters

1. messageHash - String|Object: Either signed message already prefixed and hashed or Message object with the following values:
    - message - String: The given message.
    - messageHash - String: The hash of the given message.
    - r - String: First 32 bytes of the signature.
    - s - String: Next 32 bytes of the signature.
    - v - String: Recovery value + 27.
    - signature - String: The raw RLP encoded signature.
2. signature - String: The raw RLP encoded signature.

Returns

String: The address used to sign the given message.

Example

```
eip191Signer.recover(
  'Hello World',

'0x1eab2de0103b8e82650f9706b17cf2adce55a335e7041bad5a94ab49c56a9c12662e80a369ffa2a6a77fbea
ad1f32653cbd74860c8fbc999b1fc47b8d1cb7d931c',
);
// 0x4C58e78663CB5D2Bd84Dc10beDe82A7C83442a8d;
```

</file>

<file>

path: /docs/getting-started.md

url: https://github.com/lukso-network/tools-eip191-signer/blob/main/docs/getting-started.md

---

sidebar_position: 1

Getting Started

- GitHub Repository (https://github.com/lukso-network/tools-eip191-signer)
- NPM Package (https://www.npmjs.com/package/@lukso/eip191-signer.js)

The @lukso/eip191-signer.js package is used to sign any EIP191 data.

The following format is used to sign data :

```
0x19 <1 byte version> <version specific data> <data to sign>
```

In the case of an Ethereum Signed Message:

- 1 byte version = 0x45
- version specific data = thereum Signed Message:\n + len(message)

In the case of data with intended validator:

- 1 byte version = 0x00
- version specific data = validatorAddress

  This prefix is used so that a transaction cannot be inadvertently signed when signing an Ethereum signed message.

Install

```
npm install @lukso/eip191-signer.js
```

Setup

```
import { EIP191Signer } from '@lukso/eip191-signer.js';

const eip191Signer = new EIP191Signer();
```

</file>

<file>

path: /jest.config.ts

url: https://github.com/lukso-network/tools-eip191-signer/blob/main/jest.config.ts

```
module.exports = async () => {
  return {
    roots: ['<rootDir>'],
    transform: {
      '^.+\\.tsx?{{content}}#x27;: [
        'esbuild-jest-transform',
        {
          sourcemap: true,
        },
      ],
    },
    collectCoverageFrom: ['src/**/*.ts'],
    coveragePathIgnorePatterns: [
      'node_modules',
      'interfaces',
      '<rootDir>/src/index.ts',
      '.mock.ts',
    ],
    coverageDirectory: '<rootDir>/coverage/',
    logHeapUsage: true,
    testEnvironment: 'node',
    resetMocks: true,
    maxWorkers: '50%',
    maxConcurrency: 10,
    reporters: ['default'],
  };
};
```

</file>

<file>

path: /modules.d.ts

url: https://github.com/lukso-network/tools-eip191-signer/blob/main/modules.d.ts

```
declare module 'eth-lib/lib/account';
```

</file>

<file>

path: /package.json

url: https://github.com/lukso-network/tools-eip191-signer/blob/main/package.json

```
{
  "name": "@lukso/eip191-signer.js",
  "version": "0.2.2",
```

```
"description": "Helper Library to allows to sign any EIP191 data",
"main": "build/main/src/index.js",
"typings": "build/main/src/index.d.ts",
"module": "build/module/src/index.js",
"files": [
  "build"
],
"scripts": {
  "test": "jest",
  "lint": "eslint src --ext .ts",
  "lint-fix": "eslint src --ext .ts --fix",
  "build": "run-p build:*",
  "build:main": "tsc -p tsconfig.json",
  "build:module": "tsc -p tsconfig.module.json",
  "release": "release-please",
  "contributors:add": "all-contributors add",
  "contributors:generate": "all-contributors generate"
},
"repository": {
  "type": "git",
  "url": "git+https://github.com/lukso-network/tools-eip191-signer.git"
},
"keywords": [
  "web3",
  "lukso",
  "eip191"
],
"author": {
  "name": "Magali Morin",
  "url": "https://github.com/magalimorin18"
},
"license": "Apache-2.0",
"licenses": [
  {
    "type": "Apache-2.0",
    "url": "http://www.apache.org/licenses/LICENSE-2.0"
  }
],
"bugs": {
  "url": "https://github.com/lukso-network/tools-eip191-signer/issues"
},
"homepage": "https://github.com/lukso-network/tools-eip191-signer#readme",
"devDependencies": {
  "@types/jest": "^29.0.3",
  "@types/node": "^18.7.18",
  "@typescript-eslint/eslint-plugin": "^5.37.0",
  "@typescript-eslint/parser": "^5.37.0",
  "all-contributors-cli": "^6.21.0",
  "esbuild": "^0.15.7",
  "esbuild-jest-transform": "^1.1.0",
  "eslint": "^8.23.1",
  "eslint-config-prettier": "^8.3.0",
```

```
    "eslint-plugin-eslint-comments": "^3.2.0",
    "eslint-plugin-import": "^2.26.0",
    "eslint-plugin-jest": "^27.0.4",
    "eslint-plugin-prettier": "^4.0.0",
    "jest": "^29.0.2",
    "npm-run-all": "^4.1.5",
    "prettier": "^2.3.2",
    "release-please": "^14.7.0",
    "ts-jest": "^29.0.1",
    "ts-node": "^10.9.1",
    "typescript": "^4.8.3",
    "typescript-compiler": "^1.4.1-2"
  },
  "dependencies": {
    "eth-lib": "^0.1.29",
    "ethereumjs-account": "^3.0.0",
    "ethereumjs-util": "^7.1.5",
    "web3-utils": "^1.7.5"
  }
}
```

</file>

<file>

path: /src/index.ts

url: https://github.com/lukso-network/tools-eip191-signer/blob/main/src/index.ts

```
/*
 This file contains functions to sign message for executeRelayCall.
*/
import Account from 'eth-lib/lib/account';
import { bufferToHex, keccak256 } from 'ethereumjs-util';
import utils from 'web3-utils';

import { Message } from './interfaces';

export class EIP191Signer {
  hashEthereumSignedMessage(message: string) {
    const messageHex = utils.isHexStrict(message)
      ? message
      : utils.utf8ToHex(message);
    const messageBytes = utils.hexToBytes(messageHex);
    const messageBuffer = Buffer.from(messageBytes);
    const preamble =
      '\x19' + '\x45' + 'thereum Signed Message:\n' + messageBytes.length;
    const preambleBuffer = Buffer.from(preamble);
    const ethMessage = Buffer.concat([preambleBuffer, messageBuffer]);
    return bufferToHex(keccak256(ethMessage));
  }
```

```
hashDataWithIntendedValidator(validatorAddress: string, data: string) {
  // validator address
  if (!utils.isAddress(validatorAddress))
    throw new Error('Validator needs to be a valid address');

  const validatorBuffer = Buffer.from(utils.hexToBytes(validatorAddress));
  // data to sign
  const dataHex = utils.isHexStrict(data) ? data : utils.utf8ToHex(data);
  const dataBuffer = Buffer.from(utils.hexToBytes(dataHex));

  // concatenate it
  const preambleBuffer = Buffer.from('\x19');
  const versionBuffer = Buffer.from('\x00');
  const ethMessage = Buffer.concat([
    preambleBuffer,
    versionBuffer,
    validatorBuffer,
    dataBuffer,
  ]);
  return bufferToHex(keccak256(ethMessage));
}

signEthereumSignedMessage(message: string, privateKey: string): Message {
  if (!privateKey.startsWith('0x')) {
    privateKey = '0x' + privateKey;
  }

  // 64 hex characters + hex-prefix
  if (privateKey.length !== 66) {
    throw new Error('Private key must be 32 bytes long');
  }

  const hash = this.hashEthereumSignedMessage(message);
  const signature = Account.sign(hash, privateKey);
  const vrs = Account.decodeSignature(signature);
  return {
    message: message,
    messageHash: hash,
    v: vrs[0],
    r: vrs[1],
    s: vrs[2],
    signature: signature,
  };
}

signDataWithIntendedValidator(
  validatorAddress: string,
  data: string,
  privateKey: string,
): Message {
  if (!privateKey.startsWith('0x')) {
```

```
    privateKey = '0x' + privateKey;
  }

  // 64 hex characters + hex-prefix
  if (privateKey.length !== 66) {
    throw new Error('Private key must be 32 bytes long');
  }
  const hash = this.hashDataWithIntendedValidator(validatorAddress, data);
  const signature = Account.sign(hash, privateKey);
  const vrs = Account.decodeSignature(signature);

  return {
    message: data,
    messageHash: hash,
    v: vrs[0],
    r: vrs[1],
    s: vrs[2],
    signature: signature,
  };
}

recover(messageHash: string | Message, signature: string): string {
  if (!!messageHash && typeof messageHash === 'object') {
    return this.recover(
      messageHash.messageHash,
      Account.encodeSignature([messageHash.v, messageHash.r, messageHash.s]),
    );
  }
  return Account.recover(messageHash, signature);
}
}
```

</file>

<file>

path: /src/interfaces.ts

url: https://github.com/lukso-network/tools-eip191-signer/blob/main/src/interfaces.ts

```
export interface Message {
  message: string;
  messageHash: string;
  v: string;
  r: string;
  s: string;
  signature: string;
}
```

</file>

<file>

path: /test/index.test.ts

url: https://github.com/lukso-network/tools-eip191-signer/blob/main/test/index.test.ts

```
import utils from 'web3-utils';

import { EIP191Signer } from '../src/index';

const signingKey =
  'ffeb17b9a6059fec3bbab63d76b060b7380cac7a62ce6621a134531a46458968';

const signingAddress = '0x2b389f8EB52D16A105e02165a2AC1450461A237b';

const validatorAddress = '0xAd278a6eAd89f6B6c6Fdf54A3E6E876660593b45';

const testCases = [
  'Hello World',
  '',
  ' ',
  'a',
  'https://web3js.readthedocs.io/en/v1.2.11/web3-utils.html#hextoascii',
  'mm-ll',
  '1234567890',
  '.[]{}()<>*+-=!?^$|@%',
  '0x2b389f8EB52D16A105e02165a2AC1450461A237b',
  'The family’s excitement over going to Disneyland was crazier than she anticipated.',
  'čžíáýùûüÿàâæçéèêëïîôœ',
];

const eip191Signer = new EIP191Signer();

describe('function hashEthereumSignedMessage', () => {
  it('should be hexadecimal and of 32 bites', () => {
    testCases.forEach((data) => {
      const hashedData = eip191Signer.hashEthereumSignedMessage(data);

      expect(utils.isHexStrict(hashedData)).toBeTruthy();
      expect(hashedData.length).toBe(66);
    });
  });

  it('should be prefixed with "\x19\x45thereum Signed Message:\n"', () => {
    const hashedMessage = eip191Signer.hashEthereumSignedMessage('Hello World');

    expect(hashedMessage).toBe(
      '0xa1de988600a42c4b4ab089b619297c17d53cffae5d5120d82d8a92d0bb3b78f2',
    );
  });
});
```

```
describe('function hashDataWithIntendedValidator', () => {
  it('should be hexadecimal and of 32 bites', () => {
    testCases.forEach((data) => {
      const hashedData = eip191Signer.hashDataWithIntendedValidator(
        validatorAddress,
        data,
      );

      expect(utils.isHexStrict(hashedData)).toBeTruthy();
      expect(hashedData.length).toBe(66);
    });
  });

  it('should be prefixed with "\x19\x000xAd278a6eAd89f6B6c6Fdf54A3E6E876660593b45"', () => {
    const hashedMessage = eip191Signer.hashDataWithIntendedValidator(
      validatorAddress,
      'Hello World',
    );

    expect(hashedMessage).toBe(
      '0xa63022286ecaa3317625e319a64b3bf01c41da558dfc1890e8cb196eb414ffd5',
    );
  });

  it("should throw Error with message 'Validator needs to be a valid address' when validator's address is
invalid", () => {
    testCases.forEach((data) => {
      const invalidValidatorAddress =
        '0xC1912fEE45d61C87Cc5EA59DaE31190FFFFf232d';

      function functionValidator() {
        eip191Signer.hashDataWithIntendedValidator(
          invalidValidatorAddress,
          data,
        );
      }

      expect(functionValidator).toThrow(
        new Error('Validator needs to be a valid address'),
      );
    });
  });
});

describe('function signEthereumSignedMessage', () => {
  it('should return an object with the expected properties', () => {
    testCases.forEach((data) => {
      const signedObject = eip191Signer.signEthereumSignedMessage(
        data,
        signingKey,
      );
```

```
      expect(signedObject).toHaveProperty('message');
      expect(signedObject).toHaveProperty('messageHash');
      expect(signedObject).toHaveProperty('v');
      expect(signedObject).toHaveProperty('r');
      expect(signedObject).toHaveProperty('s');
      expect(signedObject).toHaveProperty('signature');
    });
  });

  it('should sign correctly "Hello World"', () => {
    const signedObject = eip191Signer.signEthereumSignedMessage(
      'Hello World',
      signingKey,
    );

    expect(signedObject.signature).toBe(

'0x85c15865f2909897c1be6d66c1d9c86d6125978aec9e28d1a69d4d306bde694f7cf9723f0eeaf8815e3fa984ac1d7bf3c420786ead91abd4dd9c1657897efec11c',
    );
  });
});

describe('function signDataWithIntendedValidator', () => {
  it('should return an object with the expected properties', () => {
    testCases.forEach((data) => {
      const signedObject = eip191Signer.signDataWithIntendedValidator(
        validatorAddress,
        data,
        signingKey,
      );

      expect(signedObject).toHaveProperty('message');
      expect(signedObject).toHaveProperty('messageHash');
      expect(signedObject).toHaveProperty('v');
      expect(signedObject).toHaveProperty('r');
      expect(signedObject).toHaveProperty('s');
      expect(signedObject).toHaveProperty('signature');
    });
  });
  it('should sign correctly "Hello World"', () => {
    const signedObject = eip191Signer.signDataWithIntendedValidator(
      validatorAddress,
      'Hello World',
      signingKey,
    );

    expect(signedObject.signature).toBe(

'0xa7572d888a22711e180df23cf0d11748fcc0c08c0178cd88aecd1ce47b01c26469d4a87cefb20495ed07a76b4f0e4f553e32fb6333b6a325a442aae249b703181b',
```

```
    );
  });
});

describe('Recover the address of a signed EthereumSignedMessage', () => {
  it('should recover the signing address', () => {
    testCases.forEach((data) => {
      const messageData = eip191Signer.signEthereumSignedMessage(
        data,
        signingKey,
      );
      const hasedMessage = messageData.messageHash;
      const signature = messageData.signature;
      const recoveredAddress = eip191Signer.recover(hasedMessage, signature);

      expect(recoveredAddress).toBe(signingAddress);
      expect(utils.isHexStrict(recoveredAddress)).toBeTruthy();
      expect(recoveredAddress.length).toBe(42);
    });
  });

  it('should recover the signing address when the message is an object', () => {
    testCases.forEach((data) => {
      const messageData = eip191Signer.signEthereumSignedMessage(
        data,
        signingKey,
      );
      const signature = messageData.signature;
      const recoveredAddress = eip191Signer.recover(messageData, signature);

      expect(recoveredAddress).toBe(signingAddress);
    });
  });
});

describe(' Recover the address of a signed DataWithIntendedValidator', () => {
  it('should recover the signing address', () => {
    testCases.forEach((data) => {
      const messageData = eip191Signer.signDataWithIntendedValidator(
        validatorAddress,
        data,
        signingKey,
      );
      const hasedMessage = messageData.messageHash;
      const signature = messageData.signature;
      const recoveredAddress = eip191Signer.recover(hasedMessage, signature);

      expect(recoveredAddress).toBe(signingAddress);
      expect(utils.isHexStrict(recoveredAddress)).toBeTruthy();
      expect(recoveredAddress.length).toBe(42);
    });
  });
```

```
  it('should recover the signing address when the message is an object', () => {
    testCases.forEach((data) => {
      const messageData = eip191Signer.signDataWithIntendedValidator(
        validatorAddress,
        data,
        signingKey,
      );
      const signature = messageData.signature;
      const recoveredAddress = eip191Signer.recover(messageData, signature);

      expect(recoveredAddress).toBe(signingAddress);
    });
  });
});
```

</file>

<file>

path: /tsconfig.json

url: https://github.com/lukso-network/tools-eip191-signer/blob/main/tsconfig.json

```
{
  "compilerOptions": {
    "incremental": true,
    "target": "es2017",
    "outDir": "build/main",
    "rootDir": ".",
    "moduleResolution": "node",
    "module": "commonjs",
    "declaration": true,
    "inlineSourceMap": false,
    "sourceMap": true,
    "esModuleInterop": true /* Enables emit interoperability between CommonJS and ES Modules via creation of namespace objects for all imports. Implies 'allowSyntheticDefaultImports'. */,
    "resolveJsonModule": true /* Include modules imported with .json extension. */,
    /* Additional Checks */
    "noUnusedLocals": false /* Report errors on unused locals. */,
    "noUnusedParameters": true /* Report errors on unused parameters. */,
    "noImplicitReturns": true /* Report error when not all code paths in function return a value. */,
    "noFallthroughCasesInSwitch": true /* Report errors for fallthrough cases in switch statement. */,
    /* Debugging Options */
    "traceResolution": false /* Report module resolution log messages. */,
    "listEmittedFiles": false /* Print names of generated files part of the compilation. */,
    "listFiles": false /* Print names of files part of the compilation. */,
    "pretty": true /* Stylize errors and messages using color and context. */,
    "lib": ["es2017", "dom"],
    "types": ["node", "jest"],
    "typeRoots": ["node_modules/@types", "types"]
  },
  "include": ["src/**/*.ts", "types/ethers-v5/**/*.ts"],
  "exclude": ["node_modules/**", "**/*.spec.ts"],
  "compileOnSave": false
}
```

</file>

<file>

path: /tsconfig.module.json

url: https://github.com/lukso-network/tools-eip191-signer/blob/main/tsconfig.module.json

```
{
  "extends": "./tsconfig",
  "compilerOptions": {
    "target": "es2015",
    "outDir": "build/module",
    "module": "es2015"
  }
}
```

</file>

</repository>

<repository>

lsp16-factory

overview

repository: https://github.com/skimaharvey/lsp16-factory/
userName: skimaharvey
repository: lsp16-factory
branch: main
date: 2023-11-27T10:50:57+01:00 (1701078657)

<file>

path: /README.md

url: https://github.com/skimaharvey/lsp16-factory/blob/main/README.md

Pre-requisites

```
npm i
npx hardhat compile
```

Update pvtKey variable in hardhat.config.ts.

Deploy Factory

```
npx hardhat runs scripts/deployFactory.ts --network luksoTestnet
```

Deploy Owner contract through the factory

```
npx hardhat run scripts/deployOwnerThroughFactory.ts --network luksoTestnet
```

Send funds to another address

```
npx hardhat run scripts/sendFunds.ts --network luksoTestnet
```

You can also test out the pre-eip155condition by commenting out chainId.
</file>

<file>

path: /contracts/ERC20Contract.sol

url: https://github.com/skimaharvey/lsp16-factory/blob/main/contracts/ERC20Contract.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.9;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract MyToken is ERC20, Ownable {
    constructor() ERC20("MyToken", "MTK") {}

    function mint(address to, uint256 amount) public onlyOwner {
        _mint(to, amount);
    }
}
```

</file>

<file>

path: /contracts/ERC721Contract.sol

url: https://github.com/skimaharvey/lsp16-factory/blob/main/contracts/ERC721Contract.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.9;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/utils/Counters.sol";

contract MyNFT is ERC721, Ownable {
    using Counters for Counters.Counter;

    Counters.Counter private _tokenIdCounter;

    constructor() ERC721("MyNFT", "NFT") {}

    function _baseURI() internal pure override returns (string memory) {
        return "https://picsum.photos/200";
    }

    function safeMint(address to) public onlyOwner {
        uint256 tokenId = _tokenIdCounter.current();
        _tokenIdCounter.increment();
        _safeMint(to, tokenId);
    }
}
```

</file>

<file>

path: /contracts/L16UniversalFactory.sol

url: https://github.com/skimaharvey/lsp16-factory/blob/main/contracts/L16UniversalFactory.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

// libraries
import {Create2} from "@openzeppelin/contracts/utils/Create2.sol";
import {Clones} from "@openzeppelin/contracts/proxy/Clones.sol";

// errors

/**
 * @dev reverts with this error when there is no revert reason bubbled up by the target contract when initializing
 */
error CannotInitializeContract();

/**
 * @dev reverts when msg.value sent to {deployCreate2Init} function is not equal to the sum of the
 *  initializeCalldataMsgValue  and  constructorMsgValue 
 */
error InvalidMsgValueDistribution();

/**
 * @dev UniversalFactory contract can be used to deploy normal or minimal proxy contracts (EIP-1167) using CREATE2.
 * This gives the ability to deploy the same contract at the same address on different chains.
 * If the contract has a constructor, the arguments will be part of the bytecode
 * If the contract has an  initialize  function, the parameters of this function will be included in
 * the salt to ensure that the parameters of the contract should be the same on each chain.
 *
 * When initializeCallData or the constructor includes non-crosschain parameters, the deployed contract
 * will not be recreated at the same address on another network, thus defeating the purpose of LSP16UniversalFactory.

 * Therefore, the initializeCallData and the constructor must not include any network-specific parameters,
 * such as a local non-crosschain token contract address, chain-id, etc ..
 *
 * One way to solve this problem is to set an EOA owner in the initializeCallData/constructor
 * that can later call functions that set these parameters as variables in the contract.
 *
 * This contract should be deployed using Nick's Method.
 * More information: https://weka.medium.com/how-to-send-ether-to-11-440-people-187e332566b7
 */
contract LSP16UniversalFactory {
    bytes private constant _EMPTY_BYTE = "";

    /**
     * @dev Emitted whenever a contract is created
```

```
     * @param contractCreated The address of the contract created
     * @param providedSalt The bytes32 salt provided by the deployer
     * @param initializable The Boolean that specifies if the contract is a initializable or not
     * @param initializeCalldata The bytes provided as initializeCalldata
     */
    event ContractCreated(
        address indexed contractCreated,
        bytes32 indexed providedSalt,
        bool indexed initializable,
        bytes initializeCalldata
    );

    /**
     * @dev Returns the address where a contract will be stored if deployed via  CREATE2 . The address is
     * constructed using the parameters below. Any change in one of them will result in a new destination address.
     */
    function calculateAddress(
        bytes32 byteCodeHash,
        bytes32 providedSalt
    ) public view virtual returns (address) {
        return Create2.computeAddress(providedSalt, byteCodeHash);
    }

    /**
     * @dev Returns the address of an EIP1167 proxy contract. The address is constructed using
     *  the parameters below. Any change in one of them will result in a new destination address.
     */
    function calculateProxyAddress(
        address baseContract,
        bytes32 providedSalt,
        bool initializable,
        bytes calldata initializeCallData
    ) public view virtual returns (address) {
        bytes32 generatedSalt = generateSalt(initializable, initializeCallData, providedSalt);
        return Clones.predictDeterministicAddress(baseContract, generatedSalt);
    }

    /**
     * @dev Deploys a contract using  CREATE2 . The address where the contract will be deployed
     * can be known in advance via {calculateAddress}. The salt is a hash of the  providedSalt 
     * together with an empty byte, to prevent mimicing the  deployCreate2Init()  and other functions.
     *
     * This method allow users to have the same contracts at the same address across different
     * chains with the same parameters.
     *
     * Using the same  byteCode  and salt multiple time will revert, since
     * the contract cannot be deployed twice at the same address.
     */
```

```
    function deployCreate2(bytes calldata byteCode, bytes32 providedSalt)
        public
        payable
        virtual
        returns (address)
    {
        bytes32 generatedSalt = generateSalt(false, _EMPTY_BYTE, providedSalt);
        address contractCreated = Create2.deploy(msg.value, generatedSalt, byteCode);
        emit ContractCreated(contractCreated, providedSalt, false, _EMPTY_BYTE);

        return contractCreated;
    }

    /**
     * @dev Deploys a contract using  CREATE2 . The address where the contract will be
deployed
     * can be known in advance via {calculateAddress}. The salt is a hash of the  providedSalt 
and
     * and the  initializeCallData .
     *
     * This method allow users
     * to have the same contracts at the same address across different chains with the same parameters.
     *
     * The msg.value is split according to the parameters of the function
     *
     * The msg.value sent to this contract MUST be the sum of the two parameters:
 constructorMsgValue 
     * and  initializeCalldataMsgValue 
     *
     * Using the same  byteCode  and salt multiple time will revert, since
     * the contract cannot be deployed twice at the same address.
     */
    function deployCreate2Init(
        bytes calldata byteCode,
        bytes32 providedSalt,
        bytes calldata initializeCalldata,
        uint256 constructorMsgValue,
        uint256 initializeCalldataMsgValue
    ) public payable virtual returns (address) {
        if (constructorMsgValue + initializeCalldataMsgValue != msg.value)
            revert InvalidMsgValueDistribution();

        bytes32 generatedSalt = generateSalt(true, initializeCalldata, providedSalt);
        address contractCreated = Create2.deploy(constructorMsgValue, generatedSalt, byteCode);
        emit ContractCreated(contractCreated, providedSalt, true, initializeCalldata);

        (bool success, bytes memory returndata) = contractCreated.call{
            value: initializeCalldataMsgValue
        }(initializeCalldata);
        _verifyCallResult(success, returndata);

        return contractCreated;
```

```
    }

    /**
     * @dev Deploys and returns the address of a clone that mimics the behaviour of
&amp;nbsp;baseContract&amp;nbsp;.
     * The address where the contract will be deployed can be known in advance via
{calculateProxyAddress}.
     *
     * This function uses the CREATE2 opcode and a salt to deterministically deploy
     * the clone. The salt is a hash of the &amp;nbsp;providedSalt&amp;nbsp;
     * toegther with an empty byte, to prevent mimicing the &amp;nbsp;deployCreate2ProxyInit()&amp;nbsp; and other
functions.
     *
     * This method allow users to have the same contracts at the same address across different
     * chains with the same parameters.
     *
     * Using the same &amp;nbsp;baseContract&amp;nbsp; and salt multiple time will revert, since
     * the clones cannot be deployed twice at the same address.
     */
    function deployCreate2Proxy(address baseContract, bytes32 providedSalt)
        public
        virtual
        returns (address)
    {
        bytes32 generatedSalt = generateSalt(false, _EMPTY_BYTE, providedSalt);

        address proxy = Clones.cloneDeterministic(baseContract, generatedSalt);
        emit ContractCreated(proxy, providedSalt, false, _EMPTY_BYTE);

        return proxy;
    }

    /**
     * @dev Deploys and returns the address of a clone that mimics the behaviour of
&amp;nbsp;baseContract&amp;nbsp;.
     * The address where the contract will be deployed can be known in advance via
{calculateProxyAddress}.
     *
     * This function uses the CREATE2 opcode and a salt to deterministically deploy
     * the clone. The salt is a hash of the &amp;nbsp;providedSalt&amp;nbsp; and
     * and the &amp;nbsp;initializeCallData&amp;nbsp;.
     *
     * This method allow users to have the same contracts at the same address
     * across different chains with the same parameters.
     *
     * Using the same &amp;nbsp;baseContract&amp;nbsp; and salt multiple time will revert, since
     * the clones cannot be deployed twice at the same address.
     */
    function deployCreate2ProxyInit(
        address baseContract,
        bytes32 providedSalt,
        bytes calldata initializeCalldata
```

```
    ) public payable virtual returns (address) {
        bytes32 generatedSalt = generateSalt(true, initializeCalldata, providedSalt);

        address proxy = Clones.cloneDeterministic(baseContract, generatedSalt);
        emit ContractCreated(proxy, providedSalt, true, initializeCalldata);

        (bool success, bytes memory returndata) = proxy.call{value: msg.value}(initializeCalldata);
        _verifyCallResult(success, returndata);

        return proxy;
    }

    /**
     * @dev Calculates the salt used to deploy the contract by hashing (Keccak256) the following parameters
     * as packed encoded respectively: an initializable boolean, the initializeCallData if and only if
     * the contract is initializable, and the salt provided by the deployer.
     *
     * The initializable boolean was added before the provided arguments as if it was not used,
     * and we are deploying proxies on another chain, people can use the  deployCreate2(..) 
function
     * to deploy the same bytecode + the same salt to get the same address of the contract on
     * another chain without applying the effect of initializing.
     */
    function generateSalt(
        bool initializable,
        bytes memory initializeCallData,
        bytes32 providedSalt
    ) public pure virtual returns (bytes32) {
        if (initializable) {
            return keccak256(abi.encodePacked(initializable, initializeCallData, providedSalt));
        } else {
            return keccak256(abi.encodePacked(initializable, providedSalt));
        }
    }

    /**
     * @dev Verifies that the contract created was initialized correctly
     * Bubble the revert reason if present, revert with  CannotInitializeContract  otherwise
     */
    function _verifyCallResult(bool success, bytes memory returndata) internal pure virtual {
        if (!success) {
            // Look for revert reason and bubble it up if present
            if (returndata.length != 0) {
                // The easiest way to bubble the revert reason is using memory via assembly
                // solhint-disable no-inline-assembly
                /// @solidity memory-safe-assembly
                assembly {
                    let returndata_size := mload(returndata)
                    revert(add(32, returndata), returndata_size)
                }
            } else {
                revert CannotInitializeContract();
```

```
        }
      }

    }


}
```

</file>

<file>

path: /contracts/LYXeMock.sol

url: https://github.com/skimaharvey/lsp16-factory/blob/main/contracts/LYXeMock.sol

```
/**
 *Submitted for verification at Etherscan.io on 2020-05-13
 */

/**
 *Submitted for verification at Etherscan.io on 2020-05-13
 */
/*
 * source       https://github.com/lukso-network/rICO-smart-contracts
 * @name       LUKSO Token
 * @author     Micky Socaci <micky@binarzone.com>, Fabian Vogelsteller <@frozeman>
 * @license    Apachae 2.0
 */

/**
 * @dev Implementation of the  IERC777  interface.
 *
 * This implementation is agnostic to the way tokens are created. This means
 * that a supply mechanism has to be added in a derived contract using  _mint .
 *
 * Support for ERC20 is included in this contract, as specified by the EIP: both
 * the ERC777 and ERC20 interfaces can be safely used when interacting with it.
 * Both  IERC777.Sent  and  IERC20.Transfer  events are emitted on token
 * movements.
 *
 * Additionally, the  granularity  value is hard-coded to  1 , meaning that there
 * are no special restrictions in the amount of tokens that created, moved, or
 * destroyed. This makes integration with ERC20 applications seamless.
 */

pragma solidity ^0.5.0;

interface IERC777 {
    function name() external view returns (string memory);
```

```
function symbol() external view returns (string memory);

function granularity() external view returns (uint256);

function totalSupply() external view returns (uint256);

function balanceOf(address owner) external view returns (uint256);

function send(
    address recipient,
    uint256 amount,
    bytes calldata data
) external;

function burn(uint256 amount, bytes calldata data) external;

function isOperatorFor(address operator, address tokenHolder) external view returns (bool);

function authorizeOperator(address operator) external;

function revokeOperator(address operator) external;

function defaultOperators() external view returns (address[] memory);

function operatorSend(
    address sender,
    address recipient,
    uint256 amount,
    bytes calldata data,
    bytes calldata operatorData
) external;

function operatorBurn(
    address account,
    uint256 amount,
    bytes calldata data,
    bytes calldata operatorData
) external;

event Sent(
    address indexed operator,
    address indexed from,
    address indexed to,
    uint256 amount,
    bytes data,
    bytes operatorData
);

event Minted(
    address indexed operator,
    address indexed to,
    uint256 amount,
```

```
        bytes data,
        bytes operatorData
    );

    event Burned(
        address indexed operator,
        address indexed from,
        uint256 amount,
        bytes data,
        bytes operatorData
    );

    event AuthorizedOperator(address indexed operator, address indexed tokenHolder);

    event RevokedOperator(address indexed operator, address indexed tokenHolder);
}

interface IERC777Recipient {
    function tokensReceived(
        address operator,
        address from,
        address to,
        uint256 amount,
        bytes calldata userData,
        bytes calldata operatorData
    ) external;
}

interface IERC777Sender {
    function tokensToSend(
        address operator,
        address from,
        address to,
        uint256 amount,
        bytes calldata userData,
        bytes calldata operatorData
    ) external;
}

interface IERC20 {
    function totalSupply() external view returns (uint256);

    function balanceOf(address account) external view returns (uint256);

    function transfer(address recipient, uint256 amount) external returns (bool);

    function allowance(address owner, address spender) external view returns (uint256);

    function approve(address spender, uint256 amount) external returns (bool);

    function transferFrom(
        address sender,
```

```
        address recipient,
        uint256 amount
    ) external returns (bool);

    event Transfer(address indexed from, address indexed to, uint256 value);

    event Approval(address indexed owner, address indexed spender, uint256 value);
}

library SafeMath {
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow");

        return c;
    }

    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b <= a, "SafeMath: subtraction overflow");
        uint256 c = a - b;

        return c;
    }

    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b, "SafeMath: multiplication overflow");

        return c;
    }

    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b > 0, "SafeMath: division by zero");
        uint256 c = a / b;

        return c;
    }

    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b != 0, "SafeMath: modulo by zero");
        return a % b;
    }
}

library Address {
    function isContract(address account) internal view returns (bool) {
        bytes32 codehash;
        bytes32 accountHash =
```

```
0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470;

        assembly {
            codehash := extcodehash(account)
        }
        return (codehash != 0x0 && codehash != accountHash);
    }

    function toPayable(address account) internal pure returns (address payable) {
        return address(uint160(account));
    }
}

interface IERC1820Registry {
    function setManager(address account, address newManager) external;

    function getManager(address account) external view returns (address);

    function setInterfaceImplementer(
        address account,
        bytes32 interfaceHash,
        address implementer
    ) external;

    function getInterfaceImplementer(address account, bytes32 interfaceHash)
        external
        view
        returns (address);

    function interfaceHash(string calldata interfaceName) external pure returns (bytes32);

    function updateERC165Cache(address account, bytes4 interfaceId) external;

    function implementsERC165Interface(address account, bytes4 interfaceId)
        external
        view
        returns (bool);

    function implementsERC165InterfaceNoCache(address account, bytes4 interfaceId)
        external
        view
        returns (bool);

    event InterfaceImplementerSet(
        address indexed account,
        bytes32 indexed interfaceHash,
        address indexed implementer
    );

    event ManagerChanged(address indexed account, address indexed newManager);
}
```

```
contract ERC777 is IERC777, IERC20 {
    using SafeMath for uint256;
    using Address for address;

    IERC1820Registry private _erc1820 =
        IERC1820Registry(0x1820a4B7618BdE71Dce8cdc73aAB6C95905faD24);

    mapping(address => uint256) private _balances;

    uint256 private _totalSupply;

    string private _name;
    string private _symbol;

    bytes32 private constant TOKENS_SENDER_INTERFACE_HASH =
        0x29ddb589b1fb5fc7cf394961c1adf5f8c6454761adf795e67fe149f658abe895;

    bytes32 private constant TOKENS_RECIPIENT_INTERFACE_HASH =
        0xb281fc8c12954d22544db45de3159a39272895b169a852b314f9cc762e44c53b;

    address[] private _defaultOperatorsArray;

    mapping(address => bool) private _defaultOperators;

    mapping(address => mapping(address => bool)) private _operators;
    mapping(address => mapping(address => bool)) private _revokedDefaultOperators;

    mapping(address => mapping(address => uint256)) private _allowances;

    constructor(
        string memory name,
        string memory symbol,
        address[] memory defaultOperators
    ) public {
        _name = name;
        _symbol = symbol;

        _defaultOperatorsArray = defaultOperators;
        for (uint256 i = 0; i < _defaultOperatorsArray.length; i++) {
            _defaultOperators[_defaultOperatorsArray[i]] = true;
        }

        _erc1820.setInterfaceImplementer(address(this), keccak256("ERC777Token"), address(this));
        _erc1820.setInterfaceImplementer(address(this), keccak256("ERC20Token"), address(this));
    }

    function name() public view returns (string memory) {
        return _name;
    }

    function symbol() public view returns (string memory) {
        return _symbol;
```

```
}

function decimals() public pure returns (uint8) {
    return 18;
}

function granularity() public view returns (uint256) {
    return 1;
}

function totalSupply() public view returns (uint256) {
    return _totalSupply;
}

function balanceOf(address tokenHolder) public view returns (uint256) {
    return _balances[tokenHolder];
}

function send(
    address recipient,
    uint256 amount,
    bytes calldata data
) external {
    _send(msg.sender, msg.sender, recipient, amount, data, "", true);
}

function transfer(address recipient, uint256 amount) external returns (bool) {
    require(recipient != address(0), "ERC777: transfer to the zero address");

    address from = msg.sender;

    _callTokensToSend(from, from, recipient, amount, "", "");

    _move(from, from, recipient, amount, "", "");

    _callTokensReceived(from, from, recipient, amount, "", "", false);

    return true;
}

function burn(uint256 amount, bytes calldata data) external {
    _burn(msg.sender, msg.sender, amount, data, "");
}

function isOperatorFor(address operator, address tokenHolder) public view returns (bool) {
    return
        operator == tokenHolder ||
        (_defaultOperators[operator] && !_revokedDefaultOperators[tokenHolder][operator]) ||
        _operators[tokenHolder][operator];
}

function authorizeOperator(address operator) external {
```

```
    require(msg.sender != operator, "ERC777: authorizing self as operator");

    if (_defaultOperators[operator]) {
        delete _revokedDefaultOperators[msg.sender][operator];
    } else {
        _operators[msg.sender][operator] = true;
    }

    emit AuthorizedOperator(operator, msg.sender);
}

function revokeOperator(address operator) external {
    require(operator != msg.sender, "ERC777: revoking self as operator");

    if (_defaultOperators[operator]) {
        _revokedDefaultOperators[msg.sender][operator] = true;
    } else {
        delete _operators[msg.sender][operator];
    }

    emit RevokedOperator(operator, msg.sender);
}

function defaultOperators() public view returns (address[] memory) {
    return _defaultOperatorsArray;
}

function operatorSend(
    address sender,
    address recipient,
    uint256 amount,
    bytes calldata data,
    bytes calldata operatorData
) external {
    require(isOperatorFor(msg.sender, sender), "ERC777: caller is not an operator for holder");
    _send(msg.sender, sender, recipient, amount, data, operatorData, true);
}

function operatorBurn(
    address account,
    uint256 amount,
    bytes calldata data,
    bytes calldata operatorData
) external {
    require(isOperatorFor(msg.sender, account), "ERC777: caller is not an operator for holder");
    _burn(msg.sender, account, amount, data, operatorData);
}

function allowance(address holder, address spender) public view returns (uint256) {
    return _allowances[holder][spender];
}
```

```
function approve(address spender, uint256 value) external returns (bool) {
    address holder = msg.sender;
    _approve(holder, spender, value);
    return true;
}

function transferFrom(
    address holder,
    address recipient,
    uint256 amount
) external returns (bool) {
    require(recipient != address(0), "ERC777: transfer to the zero address");
    require(holder != address(0), "ERC777: transfer from the zero address");

    address spender = msg.sender;

    _callTokensToSend(spender, holder, recipient, amount, "", "");

    _move(spender, holder, recipient, amount, "", "");
    _approve(holder, spender, _allowances[holder][spender].sub(amount));

    _callTokensReceived(spender, holder, recipient, amount, "", "", false);

    return true;
}

function _mint(
    address operator,
    address account,
    uint256 amount,
    bytes memory userData,
    bytes memory operatorData
) internal {
    require(account != address(0), "ERC777: mint to the zero address");

    _totalSupply = _totalSupply.add(amount);
    _balances[account] = _balances[account].add(amount);

    _callTokensReceived(operator, address(0), account, amount, userData, operatorData, true);

    emit Minted(operator, account, amount, userData, operatorData);
    emit Transfer(address(0), account, amount);
}

function _send(
    address operator,
    address from,
    address to,
    uint256 amount,
    bytes memory userData,
    bytes memory operatorData,
    bool requireReceptionAck
```

```
) private {
    require(from != address(0), "ERC777: send from the zero address");
    require(to != address(0), "ERC777: send to the zero address");

    _callTokensToSend(operator, from, to, amount, userData, operatorData);

    _move(operator, from, to, amount, userData, operatorData);

    _callTokensReceived(
        operator,
        from,
        to,
        amount,
        userData,
        operatorData,
        requireReceptionAck
    );
}

function _burn(
    address operator,
    address from,
    uint256 amount,
    bytes memory data,
    bytes memory operatorData
) internal {
    require(from != address(0), "ERC777: burn from the zero address");

    _callTokensToSend(operator, from, address(0), amount, data, operatorData);

    _totalSupply = _totalSupply.sub(amount);
    _balances[from] = _balances[from].sub(amount);

    emit Burned(operator, from, amount, data, operatorData);
    emit Transfer(from, address(0), amount);
}

function _move(
    address operator,
    address from,
    address to,
    uint256 amount,
    bytes memory userData,
    bytes memory operatorData
) internal {
    _balances[from] = _balances[from].sub(amount);
    _balances[to] = _balances[to].add(amount);

    emit Sent(operator, from, to, amount, userData, operatorData);
    emit Transfer(from, to, amount);
}
```

```
    function _approve(
        address holder,
        address spender,
        uint256 value
    ) private {
        require(spender != address(0), "ERC777: approve to the zero address");

        _allowances[holder][spender] = value;
        emit Approval(holder, spender, value);
    }

    function _callTokensToSend(
        address operator,
        address from,
        address to,
        uint256 amount,
        bytes memory userData,
        bytes memory operatorData
    ) private {
        address implementer = _erc1820.getInterfaceImplementer(from,
TOKENS_SENDER_INTERFACE_HASH);
        if (implementer != address(0)) {
            IERC777Sender(implementer).tokensToSend(
                operator,
                from,
                to,
                amount,
                userData,
                operatorData
            );
        }
    }

    function _callTokensReceived(
        address operator,
        address from,
        address to,
        uint256 amount,
        bytes memory userData,
        bytes memory operatorData,
        bool requireReceptionAck
    ) private {
        address implementer = _erc1820.getInterfaceImplementer(to,
TOKENS_RECIPIENT_INTERFACE_HASH);
        if (implementer != address(0)) {
            IERC777Recipient(implementer).tokensReceived(
                operator,
                from,
                to,
                amount,
                userData,
                operatorData
```

```
            );
        } else if (requireReceptionAck) {
            require(
                !to.isContract(),
                "ERC777: token recipient contract has no implementer for ERC777TokensRecipient"
            );
        }
    }
}

interface ReversibleICO {
    function getParticipantReservedTokens(address) external view returns (uint256);
}

contract ReversibleICOToken is ERC777 {
    ReversibleICO public rICO;

    bool public frozen;
    bool public initialized;

    address public deployingAddress;
    address public tokenGenesisAddress;
    address public migrationAddress;
    address public freezerAddress;
    address public rescuerAddress;

    event SetRICOaddress(address indexed rICOAddress);
    event SetMigrationAddress(address indexed migrationAddress);
    event Frozen(address indexed freezerAddress);
    event Unfrozen(address indexed freezerAddress);
    event RemovedFreezer(address indexed freezerAddress);
    event ChangedRICO(address indexed rICOAddress, address indexed rescuerAddress);

    constructor(
        string memory name,
        string memory symbol,
        address[] memory _defaultOperators
    ) public ERC777(name, symbol, _defaultOperators) {
        deployingAddress = msg.sender;
    }

    function init(
        address _ricoAddress,
        address _freezerAddress,
        address _rescuerAddress,
        address _tokenGenesisAddress,
        uint256 _initialSupply
    ) public isNotInitialized onlyDeployingAddress {
        require(_freezerAddress != address(0), "_freezerAddress cannot be 0x");
        require(_rescuerAddress != address(0), "_rescuerAddress cannot be 0x");
        require(_tokenGenesisAddress != address(0), "_tokenGenesisAddress cannot be 0x");
```

```
    tokenGenesisAddress = _tokenGenesisAddress;
    freezerAddress = _freezerAddress;
    rescuerAddress = _rescuerAddress;

    _mint(_tokenGenesisAddress, _tokenGenesisAddress, _initialSupply, "", "");

    if (_ricoAddress != address(0)) {
        rICO = ReversibleICO(_ricoAddress);
        emit SetRICOaddress(_ricoAddress);
    }

    initialized = true;
}

function setRICOaddress(address _ricoAddress) public onlyTokenGenesisAddress {
    require(address(rICO) == address(0), "rICO address already set!");
    require(_ricoAddress != address(0), "rICO address cannot be 0x.");

    rICO = ReversibleICO(_ricoAddress);
    emit SetRICOaddress(_ricoAddress);
}

function setMigrationAddress(address _migrationAddress) public onlyTokenGenesisAddress {
    migrationAddress = _migrationAddress;
    emit SetMigrationAddress(migrationAddress);
}

function removeFreezer() public onlyFreezerAddress isNotFrozen {
    freezerAddress = address(0);
    emit RemovedFreezer(freezerAddress);
}

function freeze() public onlyFreezerAddress {
    frozen = true;
    emit Frozen(freezerAddress);
}

function unfreeze() public onlyFreezerAddress {
    frozen = false;
    emit Unfrozen(freezerAddress);
}

function changeRICO(address _newRicoAddress) public onlyRescuerAddress isFrozen {
    rICO = ReversibleICO(_newRicoAddress);
    emit ChangedRICO(_newRicoAddress, rescuerAddress);
}

function getLockedBalance(address _owner) public view returns (uint256) {
    if (address(rICO) != address(0)) {
        return rICO.getParticipantReservedTokens(_owner);
    } else {
        return 0;
```

```
    }
}

function getUnlockedBalance(address _owner) public view returns (uint256) {
    uint256 balance = balanceOf(_owner);

    if (address(rICO) != address(0)) {
        uint256 locked = rICO.getParticipantReservedTokens(_owner);

        if (balance > 0 && locked > 0) {
            if (balance >= locked) {
                return balance.sub(locked);
            } else {
                return 0;
            }
        }
    }

    return balance;
}

function _move(
    address _operator,
    address _from,
    address _to,
    uint256 _amount,
    bytes memory _userData,
    bytes memory _operatorData
) internal isNotFrozen isInitialized {
    if (_to == address(rICO) || _to == migrationAddress) {
        require(_amount <= balanceOf(_from), "Sending failed: Insufficient funds");
    } else {
        require(_amount <= getUnlockedBalance(_from), "Sending failed: Insufficient funds");
    }

    ERC777._move(_operator, _from, _to, _amount, _userData, _operatorData);
}

function _burn(
    address _operator,
    address _from,
    uint256 _amount,
    bytes memory _data,
    bytes memory _operatorData
) internal isNotFrozen isInitialized {
    require(_amount <= getUnlockedBalance(_from), "Burning failed: Insufficient funds");
    ERC777._burn(_operator, _from, _amount, _data, _operatorData);
}

modifier onlyDeployingAddress() {
    require(msg.sender == deployingAddress, "Only the deployer can call this method.");
    _;
```

```
    }

    modifier onlyTokenGenesisAddress() {
        require(
            msg.sender == tokenGenesisAddress,
            "Only the tokenGenesisAddress can call this method."
        );
        _;
    }

    modifier onlyFreezerAddress() {
        require(msg.sender == freezerAddress, "Only the freezer address can call this method.");
        _;
    }

    modifier onlyRescuerAddress() {
        require(msg.sender == rescuerAddress, "Only the rescuer address can call this method.");
        _;
    }

    modifier isInitialized() {
        require(initialized == true, "Contract must be initialized.");
        _;
    }

    modifier isNotInitialized() {
        require(initialized == false, "Contract is already initialized.");
        _;
    }

    modifier isFrozen() {
        require(frozen == true, "Token contract not frozen.");
        _;
    }

    modifier isNotFrozen() {
        require(frozen == false, "Token contract is frozen!");
        _;
    }
}
```

</file>

<file>

path: /contracts/Owner.sol

url: https://github.com/skimaharvey/lsp16-factory/blob/main/contracts/Owner.sol

```
// SPDX-License-Identifier: GPL-3.0
```

```
pragma solidity >=0.7.0 <0.9.0;

/**
 * @title Owner
 * @dev Set & change owner
 */
contract Owner {

    address private owner;

    // event for EVM logging
    event OwnerSet(address indexed oldOwner, address indexed newOwner);

    // modifier to check if caller is owner
    modifier isOwner() {
        // If the first argument of 'require' evaluates to 'false', execution terminates and all
        // changes to the state and to Ether balances are reverted.
        // This used to consume all gas in old EVM versions, but not anymore.
        // It is often a good idea to use 'require' to check if functions are called correctly.
        // As a second argument, you can also provide an explanation about what went wrong.
        require(msg.sender == owner, "Caller is not owner");
        _;
    }

    /**
     * @dev Set contract deployer as owner
     */
    constructor() {
        owner = msg.sender; // 'msg.sender' is sender of current call, contract deployer for a constructor
        emit OwnerSet(address(0), owner);
    }

    function generateMappingWithGroupingKey(
        bytes6 keyPrefix,
        bytes4 mapPrefix,
        bytes20 subMapKey
    ) external pure returns (bytes32) {
        bytes memory generatedKey = bytes.concat(keyPrefix, mapPrefix, bytes2(0), subMapKey);
        return bytes32(generatedKey);
    }

    function generateMappingWithGroupingKey2(
        bytes6 keyPrefix,
        bytes4 mapPrefix,
        bytes20 subMapKey
    ) external pure returns (bytes32) {
        return bytes32(bytes.concat(keyPrefix, mapPrefix, bytes2(0), subMapKey));
    }

    /**
     * @dev Return owner address
     * @return address of owner
```

```
    */
    function getOwner() external view returns (address) {
        return owner;
    }
}
```

</file>

<file>

path: /contracts/RevertContract.sol

url: https://github.com/skimaharvey/lsp16-factory/blob/main/contracts/RevertContract.sol

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.4;

/**
 * @title Owner
 * @dev Set & change owner
 */
contract RevertContract {

   error CustomError(string message);


    function revertHere() external pure {
        revert("This is a revert message");
    }

    function requireOver1(uint256 _value) external pure {
        require(_value > 1, "Value must be over 1");
    }

    function revertCustomError() external pure {
      revert CustomError("This is a custom error message");
    }

}
```

</file>

<file>

path: /contracts/UniversalProfileDeployer.sol

url: https://github.com/skimaharvey/lsp16-factory/blob/main/contracts/UniversalProfileDeployer.sol

```
// SPDX-License-Identifier: MIT
```

```
pragma solidity ^0.8.4;

import {_LSP6KEY_ADDRESSPERMISSIONS_PERMISSIONS_PREFIX,
_PERMISSION_CHANGEOWNER, _PERMISSION_EDITPERMISSIONS,
ALL_REGULAR_PERMISSIONS} from "@lukso/lsp-smart-
contracts/contracts/LSP6KeyManager/LSP6Constants.sol";
import {LSP6Utils} from '@lukso/lsp-smart-contracts/contracts/LSP6KeyManager/LSP6Utils.sol';
import {Create2} from "@openzeppelin/contracts/utils/Create2.sol";
import {ILSP14Ownable2Step} from '@lukso/lsp-smart-
contracts/contracts/LSP14Ownable2Step/ILSP14Ownable2Step.sol';



contract UniversalProfileDeployer {

    function deployUPAndKeyManager(bytes calldata universalProfileByteCode,  bytes calldata
keyManagerByteCode, address allPermissionsAddress, bytes32 universalProfleProvidedSalt, bytes32
keyManagerProvidedSalt)
        public
        payable
        virtual
        returns (address universalProfile, address keyManager)
    {   // generate salt for the UP contract
        bytes32 universalProfileGeneratedSalt = keccak256(abi.encodePacked(allPermissionsAddress,
keyManagerByteCode, universalProfleProvidedSalt));

        // deploy UP contract
        universalProfile = Create2.deploy(msg.value, universalProfileGeneratedSalt,
abi.encodePacked(universalProfileByteCode,abi.encode(address(this))));

        // deploy keyManager contract
        keyManager = Create2.deploy(0, keyManagerProvidedSalt,
abi.encodePacked(keyManagerByteCode,abi.encode(universalProfile)));

        // calculate deployer permissions
        bytes32[] memory deployerPermissionsArray = new bytes32[](2);
        deployerPermissionsArray[0] = _PERMISSION_CHANGEOWNER;
        deployerPermissionsArray[1] = _PERMISSION_EDITPERMISSIONS;
        bytes32 deployerPermissions = LSP6Utils.combinePermissions(deployerPermissionsArray);

        // setDataBatch keys
        bytes32 deloyerKey =
bytes32(abi.encodePacked(_LSP6KEY_ADDRESSPERMISSIONS_PERMISSIONS_PREFIX, bytes2(0),
address(this)));
        bytes32 allPermissionsKey =
bytes32(abi.encodePacked(_LSP6KEY_ADDRESSPERMISSIONS_PERMISSIONS_PREFIX, bytes2(0),
allPermissionsAddress));
        bytes32[] memory keys = new bytes32[](2);
        keys[0] = deloyerKey;
        keys[1] = allPermissionsKey;

        // setDataBach values
```

```
        bytes[] memory values = new bytes[](2);
        values[0] = abi.encodePacked(deployerPermissions);
        values[1] = abi.encodePacked(ALL_REGULAR_PERMISSIONS);

        // setDataBatch on UP contract
        (bool success, ) = universalProfile.call(abi.encodeWithSignature("setDataBatch(bytes32[],bytes[])",
keys, values));
        require(success, "setDataBatch failed");

        // transferOwnership on UP contract
        (bool successTransfer, ) = universalProfile.call(abi.encodeWithSignature("transferOwnership(address)",
keyManager));
        require(successTransfer, "transferOwnership failed");

        // acceptOwnership on keyManager contract
        bytes memory acceptOwnershipBytes = abi.encodeWithSignature("acceptOwnership()");
        (bool successAccept,) = keyManager.call(abi.encodeWithSignature("execute(bytes)",
acceptOwnershipBytes));
        require(successAccept, "acceptOwnership failed");

        // setData on keyManager contract
        bytes memory setDataBytes = abi.encodeWithSignature("setData(bytes32,bytes)", deloyerKey, "");
        (bool successSetData,) = keyManager.call(abi.encodeWithSignature("execute(bytes)", setDataBytes));
        require(successSetData, "setData failed");
    }

}
```

</file>

<file>

path: /flattened.sol

url: https://github.com/skimaharvey/lsp16-factory/blob/main/flattened.sol

```
// Sources flattened with hardhat v2.14.0 https://hardhat.org

// File contracts/RevertContract.sol

// SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.4;

/**
 * @title Owner
 * @dev Set & change owner
 */
contract RevertContract {

  error CustomError(string message);


   function revertHere() external pure {
      revert("This is a revert message");
   }

   function requireOver1(uint256 _value) external pure {
      require(_value > 1, "Value must be over 1");
   }

   function revertCustomError() external pure {
     revert CustomError("This is a custom error message");
   }

}
```

</file>

<file>

path: /hardhat.config.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/hardhat.config.ts

```
import { HardhatUserConfig } from "hardhat/config";
import "@nomicfoundation/hardhat-toolbox";
import "@nomiclabs/hardhat-etherscan";

const pvtKey = '0xbbc6703446945a0d6e1d40d50664da1c37bf51a1383ce165af96ccaa62c8f56e'
const pvtKeyMainnet = '0xd8406bf2f8048957fd35c67f9f32b75516d1601b7aa8cb367a50ebd5a69bebbe'

const config: HardhatUserConfig = {
  solidity: {
    compilers: [
      {
        version: "0.5.0",
```

```
      },

      {
        version: "0.8.15"
      },
    ],
  },
  networks: {
    hardhat: {
      blockGasLimit: 32999064,
      // url: &amp;nbsp;http://localhost:8545&amp;nbsp;,
      // accounts:
[&amp;nbsp;0xac0974bec39a17e36ba4a6b4d238ff944bacb478cbed5efcae784d7bf4f2ff80&amp;nbsp;],
    },
    luksoTestnet: {
      url: &amp;nbsp;https://rpc.testnet.lukso.network&amp;nbsp;,
      accounts: [pvtKey],
    },
    luksoMainnet: {
      url: &amp;nbsp;https://rpc.mainnet.lukso.network&amp;nbsp;,
      accounts: [pvtKeyMainnet],
    },
    l3030: {
      url: &amp;nbsp;https://rpc.execution.3030.devnet.lukso.dev&amp;nbsp;,
      accounts: [pvtKey],
    },
    localhost: {
      url: &amp;nbsp;http://localhost:8545&amp;nbsp;,
      accounts: [&amp;nbsp;0xac0974bec39a17e36ba4a6b4d238ff944bacb478cbed5efcae784d7bf4f2ff80&amp;nbsp;],
    }
  },
  etherscan: {
    apiKey: 'no-api-key-needed',
    customChains: [
      {
        network: "luksoTestnet",
        chainId: 4201,
        urls: {
          apiURL: "https://explorer.execution.testnet.lukso.network/api",
          browserURL: "https://explorer.execution.testnet.lukso.network/",
        },
      },
    ],
  },
};

export default config;
```

</file>

<file>

path: /package.json

url: https://github.com/skimaharvey/lsp16-factory/blob/main/package.json

```
{
  "name": "hardhat-project",
  "devDependencies": {
    "@nomicfoundation/hardhat-toolbox": "^2.0.2",
    "@nomiclabs/hardhat-etherscan": "^3.1.7",
    "hardhat": "^2.14.0"
  },
  "dependencies": {
    "@lukso/lsp-smart-contracts": "^0.10.1",
    "@openzeppelin/contracts": "^4.8.3"
  }
}
```

</file>

<file>

path: /scripts/acceptOwnership.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/acceptOwnership.ts

```
import { ethers } from "hardhat";


const KEY_MANAGER_ADDRESS = "0xc8951DA73CE9451D0BD8eBB6537364630F8b2985"

async function main() {
  const keyManager = await ethers.getContractAt("LSP6KeyManager", KEY_MANAGER_ADDRESS)

  const [Signer] = await ethers.getSigners();
  console.log('Signer', Signer.address)

  const UniversalProfile = await ethers.getContractFactory("UniversalProfile");
  const acceptOwnershipBytecode = UniversalProfile.interface.encodeFunctionData('acceptOwnership');
  const tx = await keyManager.execute(acceptOwnershipBytecode, {gasPrice: await
ethers.provider.getGasPrice()});
  await tx.wait();

}



main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

</file>

<file>

path: /scripts/callRevert.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/callRevert.ts

```
import {ethers } from 'hardhat';









(async function() {

    try {
        const revertContractAddress = '0xd4A8623c881B183Fc49f3F766aB0a41B2a141929';
        const revertContract = await ethers.getContractAt('RevertContract', revertContractAddress);
        await revertContract.revertHere({gasPrice: await ethers.provider.getGasPrice()});

    } catch (error) {
        console.error('Error:', error);
    }
})();
```

</file>

<file>

path: /scripts/checkBytecode.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/checkBytecode.ts

```
import {ethers } from 'hardhat';




(async function() {

    try {
        const addressToCheckBytecodeFor = '0x3b7235344F490dff8A4491cF7294774b4cC337f4';
        const bytecode = await ethers.provider.getCode(addressToCheckBytecodeFor);
        console.log('Bytecode:', bytecode);
    } catch (error) {
        console.error('Error:', error);
    }
})();
```

</file>

<file>

path: /scripts/checkTransactionPool.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/checkTransactionPool.ts

```
import {ethers} from 'ethers';



const provider = new ethers.providers.JsonRpcProvider('https://rpc.execution.testnet.lukso.io');


(async function() {
  try {
    const txPoolContent = await provider.send('txpool_content', []);
    console.log('Transaction Pool Content:', JSON.stringify(txPoolContent, null, 2));
  } catch (error) {
    console.error('Error fetching transaction pool content:', error.message);
  }
})();
```

</file>

<file>

path: /scripts/deployERC20Token.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/deployERC20Token.ts

```
import { ethers } from "hardhat";

async function main() {
  const gasPrice = await ethers.provider.getGasPrice();

  const MyTokenFactory = await ethers.getContractFactory("MyToken");

  console.log( Deploying ERC20TokenContract with gas price ${await
ethers.provider.getGasPrice()}... )

  const myTokenContract = await MyTokenFactory.deploy({gasPrice});

  await myTokenContract.deployed();

  console.log(
     Deployed to ${myTokenContract.address} 
  );
}


main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

</file>

<file>

path: /scripts/deployFactory.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/deployFactory.ts

```
import { ethers } from "hardhat";

async function main() {
  const gasPrice = await ethers.provider.getGasPrice();

  const LSP16UniversalFactory = await ethers.getContractFactory("LSP16UniversalFactory");

  console.log( Deploying LSP16UniversalFactory with gas price ${await
ethers.provider.getGasPrice()}... )

  const universalFactory = await LSP16UniversalFactory.deploy();

  await universalFactory.deployed();

  console.log(
     Deployed to ${universalFactory.address} 
  );
}


main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

</file>

<file>

path: /scripts/deployKeyManager.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/deployKeyManager.ts

```
import { ethers } from "hardhat";

const UNIVERSAL_PROFILE_ADDRESS = "0x9d899301580e04495344d96a42959a5F90c59093"

async function main() {
  const KeyManagerFactory = await ethers.getContractFactory("LSP6KeyManager");
  const keyManager = await KeyManagerFactory.deploy(UNIVERSAL_PROFILE_ADDRESS,{gasPrice:
(await ethers.provider.getGasPrice()).add(ethers.utils.parseUnits("1", "gwei"))});

  await keyManager.deployed();

  console.log('keyManager', keyManager.address)
}



main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

</file>

<file>

path: /scripts/deployLYXeMock.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/deployLYXeMock.ts

```
import { ethers } from "hardhat";

async function main() {
  const LYXeMockFactory = await ethers.getContractFactory(
    "ReversibleICOToken"
  );

  const lyxeMockContract = await LYXeMockFactory.deploy("LYXe", "LYXe", []);
  await lyxeMockContract.deployed();
  console.log("lyxeMockContract deployed to:", lyxeMockContract.address);
}

main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

</file>

<file>

path: /scripts/deployNFT.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/deployNFT.ts

```
import { ethers } from "hardhat";

async function main() {
  const gasPrice = await ethers.provider.getGasPrice();

  const MyTokenFactory = await ethers.getContractFactory("MyNFT");

  console.log( Deploying NFT with gas price ${await ethers.provider.getGasPrice()}... )

  const myTokenContract = await MyTokenFactory.deploy({gasPrice});

  await myTokenContract.deployed();

  console.log(
     Deployed to ${myTokenContract.address} 
  );
}


main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

</file>

<file>

path: /scripts/deployOwnerThroughFactory.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/deployOwnerThroughFactory.ts

```
import { ethers } from "hardhat";
import {bytecode as ownerBytecode} from '../artifacts/contracts/Owner.sol/Owner.json'

async function main() {
  const gasPrice = await ethers.provider.getGasPrice();
  const universalFactoryAddress = '0x03BB0cBbc9dd38b5e7dD32e42c89fB00B61fCCB1'

  const randomSalt = ethers.utils.hexlify(ethers.utils.randomBytes(32));
  console.log(&amp;nbsp;Random salt: ${randomSalt}&amp;nbsp;)


  const universalFactory = await ethers.getContractAt("LSP16UniversalFactory", universalFactoryAddress);

  const contractAddress = await universalFactory.callStatic.deployCreate2(ownerBytecode,randomSalt,
{gasPrice});

  const tx = await universalFactory.deployCreate2(ownerBytecode,randomSalt, {gasPrice});

  await tx.wait();

  console.log(
    &amp;nbsp;Deployed to ${contractAddress}&amp;nbsp;
  );
}


main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

</file>

<file>

path: /scripts/deployRevertContract.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/deployRevertContract.ts

```
import { ethers } from "hardhat";

async function main() {
  const gasPrice = await ethers.provider.getGasPrice();

  const RevertContractFactory = await ethers.getContractFactory("RevertContract");

  console.log( Deploying RevertContract with gas price ${await
ethers.provider.getGasPrice()}... )

  const revertContract = await RevertContractFactory.deploy({gasPrice});

  await revertContract.deployed();

  console.log(
     Deployed to ${revertContract.address} 
  );
}


main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

</file>

<file>

path: /scripts/deployUP.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/deployUP.ts

```
import { ethers } from "hardhat";


async function main() {

  const owner = '0x9d899301580e04495344d96a42959a5F90c59093'

  const UPContract = await ethers.getContractFactory("UniversalProfile");
  const upContract = await UPContract.deploy(owner, {gasPrice: (await
ethers.provider.getGasPrice()).add(ethers.utils.parseUnits("1", "gwei"))});

  await upContract.deployed();

  console.log('upContract', upContract.address)


}




main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

</file>

<file>

path: /scripts/deployUPDeployer.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/deployUPDeployer.ts

```
import { ethers } from "hardhat";

async function main() {
  const UPDeployerFactory = await ethers.getContractFactory("UniversalProfileDeployer");
  const UPDeployer = await UPDeployerFactory.deploy({gasPrice: (await
ethers.provider.getGasPrice()).add(ethers.utils.parseUnits("1", "gwei"))});

  await UPDeployer.deployed();

  console.log("UPDeployer deployed to:", UPDeployer.address);
  // deployed locally at 0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512
  // deployed on Testnet at 0xbcfb13485782cf6631157e6a05B56982fe076f39
}




main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

</file>

<file>

path: /scripts/deployUPThroughDeployer.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/deployUPThroughDeployer.ts

```
import { ethers } from "hardhat";

const UP_DEPLOYER_ADDRESS = "0xC100520c25c6c51a98EFc0FB5c6553fa92D3FfdB"

async function main() {

  const UPDeployerContract = await ethers.getContractAt("UniversalProfileDeployer",
UP_DEPLOYER_ADDRESS);

  const UniversalProfile = await ethers.getContractFactory("UniversalProfile");
  const universalProfileBytecode = UniversalProfile.bytecode;

  const KeyManager = await ethers.getContractFactory("LSP6KeyManager");
  const keyManagerBytecode = KeyManager.bytecode;

  const universalProfileSalt = ethers.utils.randomBytes(32);
  const keyManagerSalt = ethers.utils.randomBytes(32);

  const [signerWithAllPermissions] = await ethers.getSigners();
  console.log('signerWithAllPermissions', signerWithAllPermissions.address)

  const estimateGas = await
UPDeployerContract.estimateGas.deployUPAndKeyManager(universalProfileBytecode,
keyManagerBytecode, signerWithAllPermissions.address, universalProfileSalt, keyManagerSalt);
  console.log('estimateGas', estimateGas.toString())

  const [universalProfile, keyManager] = await
UPDeployerContract.callStatic.deployUPAndKeyManager(universalProfileBytecode, keyManagerBytecode,
signerWithAllPermissions.address, universalProfileSalt, keyManagerSalt, {gasPrice: (await
ethers.provider.getGasPrice()).add(ethers.utils.parseUnits("1", "gwei"))});
  console.log('universalProfile address', universalProfile)
  console.log('keyManager address', keyManager)

  const tx = await UPDeployerContract.deployUPAndKeyManager(universalProfileBytecode,
keyManagerBytecode, signerWithAllPermissions.address, universalProfileSalt, keyManagerSalt, {gasPrice:
(await ethers.provider.getGasPrice()).add(ethers.utils.parseUnits("1", "gwei"))});
  await tx.wait();
  console.log('tx', tx);

}




main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

</file>

<file>

path: /scripts/fetchData.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/fetchData.ts

```
import { ethers } from "hardhat";


const UPAddress = '0xEE0Ee0668933A86BC42b80331dc6eF39D1Ed9414'
const KEY_DATA_PREFIX = '0x4b80742de2bf82acb3630000'
const UP_DEPLOYER = '0x0eCC079C20DaA9fDE0e26b6d745c0b38479ff200'

async function main() {
  const [signerWithAllPermissions] = await ethers.getSigners();
  console.log('signerWithAllPermissions', signerWithAllPermissions.address)

  const UPContract = await ethers.getContractAt("UniversalProfile", UPAddress);
  const bytes = await UPContract.callStatic.getData(KEY_DATA_PREFIX + UP_DEPLOYER.slice(2));
  console.log('bytes', bytes)


}



main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

</file>

<file>

path: /scripts/fetchTarget.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/fetchTarget.ts

```
import { ethers } from "hardhat";
const KEY_MANAGER_ADDRESS = '0xcA5B1657C30D52D75Cf52a70dcf1b87B8a57a36E'

async function main() {

  const keyManager = await ethers.getContractAt("LSP6KeyManager", KEY_MANAGER_ADDRESS);

  const target = await keyManager.callStatic.target();

  console.log('target', target)

}




main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

</file>

<file>

path: /scripts/fetchUPOwner.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/fetchUPOwner.ts

```
const UPAddress = '0xEE0Ee0668933A86BC42b80331dc6eF39D1Ed9414'

async function main() {

  const UPContract = await ethers.getContractAt("UniversalProfile", UPAddress);

  const owner = await UPContract.callStatic.owner();

  const pendingOwner = await UPContract.callStatic.pendingOwner();

  console.log('owner', owner)
  console.log('pendingOwner', pendingOwner)

}




main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

</file>

<file>

path: /scripts/mintListener.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/mintListener.ts

```
import { ethers } from "ethers";

// Connect to the network
let provider = new ethers.providers.WebSocketProvider("https://ws-rpc.testnet.lukso.network");

// The address that the Contract WILL BE deployed to when we deploy it (replace with your contract address)
let contractAddress = "0xfB10A65e5f0402b8e9D4Ca84D0F367F0779Cc67F";

// The Contract interface
let abi = [
  // ERC721 Transfer event
  "event Transfer(address indexed from, address indexed to, uint256 indexed tokenId)"
];

// We connect to the Contract using a Provider, so we will only
// have read-only access to the Contract
let contract = new ethers.Contract(contractAddress, abi, provider);

contract.on("Transfer", (from, to, tokenId, event) => {
  // This will be called when a Transfer event is emitted
  console.log( Transfer event. From: ${from}, To: ${to}, TokenId: ${tokenId.toString()} );
});

// In case of any errors
contract.on("error", (error, event) => {
  console.log(error, event);
});
```

</file>

<file>

path: /scripts/mintNFT.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/mintNFT.ts

```
import {ethers } from 'hardhat';


(async function() {

    try {
        const nftContractAddress = '0x9006dd915576ff0F44034Fe3D09fd4825dB5b6CB';
        const erc20Contract = await ethers.getContractAt('MyNFT', nftContractAddress);
        const gasPrice = await ethers.provider.getGasPrice();
        const tx = await erc20Contract.safeMint('0x4FAea6Ed258f979DfeF9e77F1dE41f3E916908f5',
{gasPrice})

        await tx.wait();

    } catch (error) {
        console.error('Error:', error);
    }
})();
```

</file>

<file>

path: /scripts/mintToken.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/mintToken.ts

```
import {ethers } from 'hardhat';


(async function() {

    try {
        const erc20TokenAddress = '0xD5F39656f163F11385f3352b2e9e1540c4DB664d';
        const erc20Contract = await ethers.getContractAt('MyToken', erc20TokenAddress);
        const gasPrice = await ethers.provider.getGasPrice();
        const tx = await erc20Contract.mint('0x4FAea6Ed258f979DfeF9e77F1dE41f3E916908f5',
ethers.utils.parseEther('1000'), {gasPrice})

        await tx.wait();

    } catch (error) {
        console.error('Error:', error);
    }
})();
```

</file>

<file>

path: /scripts/sendFunds.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/sendFunds.ts

```
import {ethers} from 'ethers';

const provider = new ethers.providers.JsonRpcProvider('https://rpc.testnet.lukso.network');

const privateKey = '0xbbc6703446945a0d6e1d40d50664da1c37bf51a1383ce165af96ccaa62c8f56e';
const wallet = new ethers.Wallet(privateKey, provider);
const walletAddress = wallet.address;

console.log('Wallet address:', walletAddress);


const recipientAddress = '0xeC21Ad25A3fB6483930C2e7BC7E8a48338B3a2BB';
const amountToSend = ethers.utils.parseEther('0.1');



(async function() {
   console.log('nonce:', await provider.getTransactionCount(walletAddress));
   const transaction = {
     to: recipientAddress,
     value: amountToSend,
     gasLimit: 50000,
     gasPrice: ethers.utils.parseUnits('4.5', 'gwei'),
     chainId: 4201, // will fail if network is not pre-eip155 enabled
     nonce: await provider.getTransactionCount(walletAddress),
 };

   const balance = await provider.getBalance(walletAddress);
   console.log('Balance:', ethers.utils.formatEther(balance));
   try {
       const signedTransaction = await wallet.signTransaction(transaction);
       const transactionResponse = await provider.sendTransaction(signedTransaction);
       console.log('Transaction hash:', transactionResponse.hash);
   } catch (error) {
       console.error('Error:', error);
   }
})();
```

</file>

<file>

path: /scripts/test_ethLogs.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/test_ethLogs.ts

```
const ethers = require('ethers');

async function getLogs() {
    const provider = new ethers.providers.JsonRpcProvider('https://rpc.testnet.lukso.network');

    const getBlockNumber = await provider.getBlockNumber();
    const startBlockNumber = getBlockNumber - 1000;
    const blockNumberToHexString = ethers.utils.hexValue(getBlockNumber);

    const blockNumberStartToHexString = ethers.utils.hexValue(startBlockNumber);

    console.log(typeof(blockNumberToHexString));

    const params = {
        fromBlock: blockNumberStartToHexString,  // Hexadecimal string for block number. Replace with your values.
        toBlock: blockNumberToHexString,
        address: '0xfB10A65e5f0402b8e9D4Ca84D0F367F0779Cc67F',  // Replace with your contract address.
        topics: []  // Array of topics. Fill with your values.
    };

    const logs = await provider.send('eth_getLogs', [params]);

    console.log(logs);
}

getLogs().catch(console.error);
```

</file>

<file>

path: /scripts/transferNFT.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/transferNFT.ts

```
import {ethers } from 'hardhat';


(async function() {

    try {
        const nftContractAddress = '0xfB10A65e5f0402b8e9D4Ca84D0F367F0779Cc67F';
        const erc20Contract = await ethers.getContractAt('MyNFT', nftContractAddress);
        const gasPrice = await ethers.provider.getGasPrice();
        const tx = await
erc20Contract.transferFrom('0x0eCC079C20DaA9fDE0e26b6d745c0b38479ff200','0x6C36b9aA78a49C107
0abff2A4975b5df40d77eAc', 0, {gasPrice})

        await tx.wait();

    } catch (error) {
        console.error('Error:', error);
    }
})();
```

</file>

<file>

path: /scripts/transferToken.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/scripts/transferToken.ts

```
import {ethers } from 'hardhat';


(async function() {

    try {
        const erc20TokenAddress = '0x8A7aCf6e006B4Ce499Eb95C9c109C5Baa59585cB';
        const erc20Contract = await ethers.getContractAt('MyToken', erc20TokenAddress);
        const gasPrice = await ethers.provider.getGasPrice();
        const tx = await erc20Contract.transfer('0x7975Bc26b6be137Af7417e8977C7966D731c88fD',
ethers.utils.parseEther('100'), {gasPrice})

    } catch (error) {
        console.error('Error:', error);
    }
})();
```

</file>

<file>

path: /test/UniversalProfileDeployer.test.ts

url: https://github.com/skimaharvey/lsp16-factory/blob/main/test/UniversalProfileDeployer.test.ts

```
import { ethers } from "hardhat";
import { expect } from "chai";
import { bytecode } from '../artifacts/@lukso/lsp-smart-
contracts/contracts/UniversalProfile.sol/UniversalProfile.json'

describe("UniversalProfileDeployer", function () {
  it("Should deploy a UniversalProfile", async function () {
    const UniversalProfileDeployer = await ethers.getContractFactory("UniversalProfileDeployer");
    const universalProfileDeployer = await UniversalProfileDeployer.deploy();
    await universalProfileDeployer.deployed();

    const UniversalProfile = await ethers.getContractFactory("UniversalProfile");
    const universalProfileBytecode = UniversalProfile.bytecode;

    const KeyManager = await ethers.getContractFactory("LSP6KeyManager");
    const keyManagerBytecode = KeyManager.bytecode;

    const generateTwelveBytesSalt = ethers.utils.randomBytes(12);
    const generateThirtyTwoBytesSalt = ethers.utils.randomBytes(32);

    const [signerWithAllPermissions] = await ethers.getSigners();
    const [universalProfile, keyManager] = await
universalProfileDeployer.callStatic.deployUPAndKeyManager(universalProfileBytecode,
keyManagerBytecode, signerWithAllPermissions.address, generateTwelveBytesSalt,
generateThirtyTwoBytesSalt);

    await universalProfileDeployer.deployUPAndKeyManager(universalProfileBytecode,
keyManagerBytecode, signerWithAllPermissions.address, generateTwelveBytesSalt,
generateThirtyTwoBytesSalt);

    const universalProfileContract = UniversalProfile.attach(universalProfile);
    const keyManagerContract = KeyManager.attach(keyManager);

    expect(await universalProfileContract.owner()).to.equal(keyManager);

  });
});
```

</file>

<file>

path: /tsconfig.json

url: https://github.com/skimaharvey/lsp16-factory/blob/main/tsconfig.json

```
{
  "compilerOptions": {
    "target": "es2020",
    "module": "commonjs",
    "esModuleInterop": true,
    "forceConsistentCasingInFileNames": true,
    "strict": true,
    "skipLibCheck": true,
    "resolveJsonModule": true
  }
}
```

</file>

</repository>