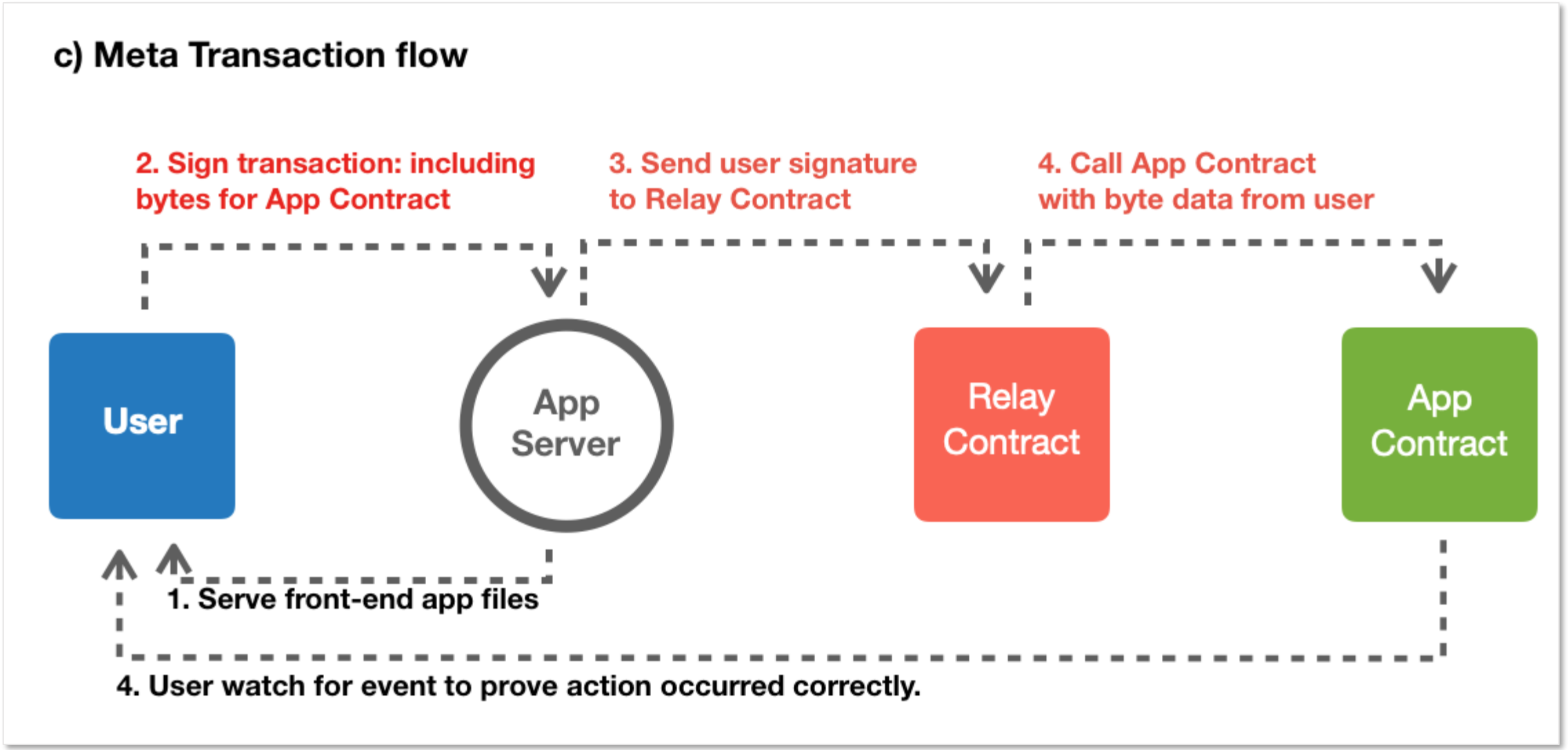
PD-15 New Developments

PD-15.1 MetaTransactions



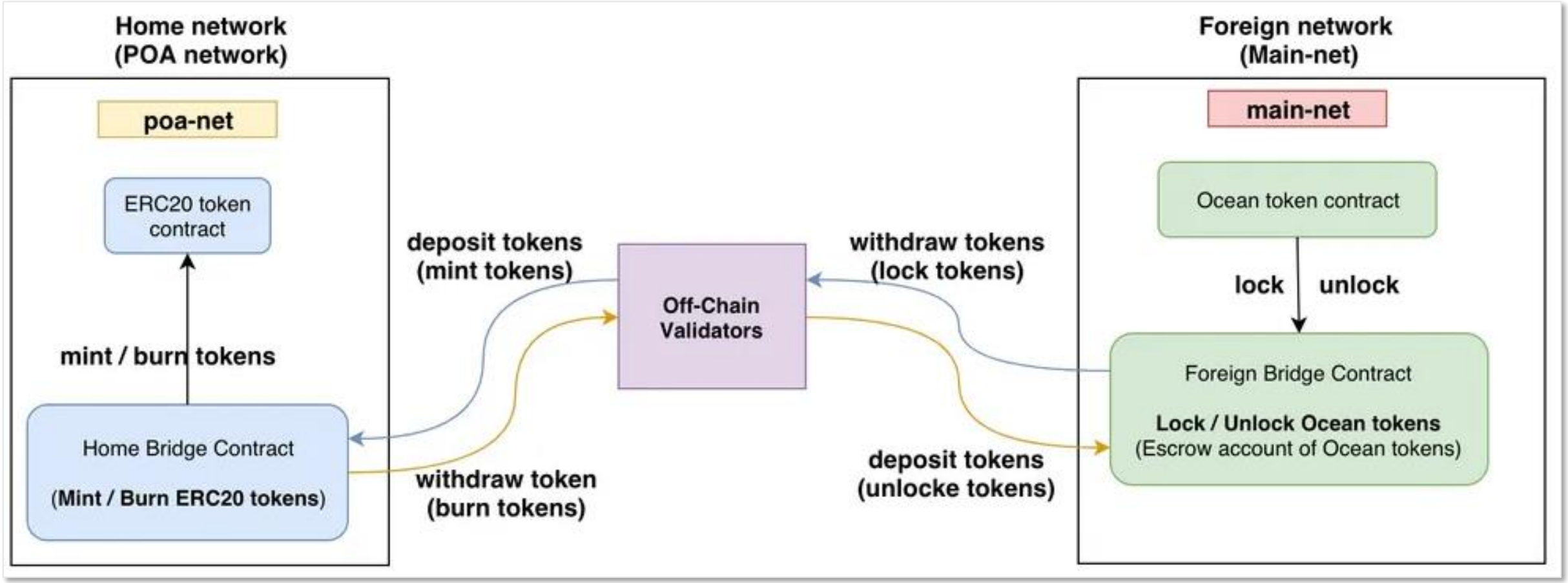
https://infura.io/docs/ethereum#tag/Transactions

https://opengsn.org

https://github.com/tsuzukit/meta-transaction

https://medium.com/@RongxinZhang/meta-transactions-gasless-transactions-7c0ce64ef9bd

PD-15.2 Bridges



https://github.com/ensdomains/l2gateway-demo

https://github.com/ChainSafe/ChainBridge

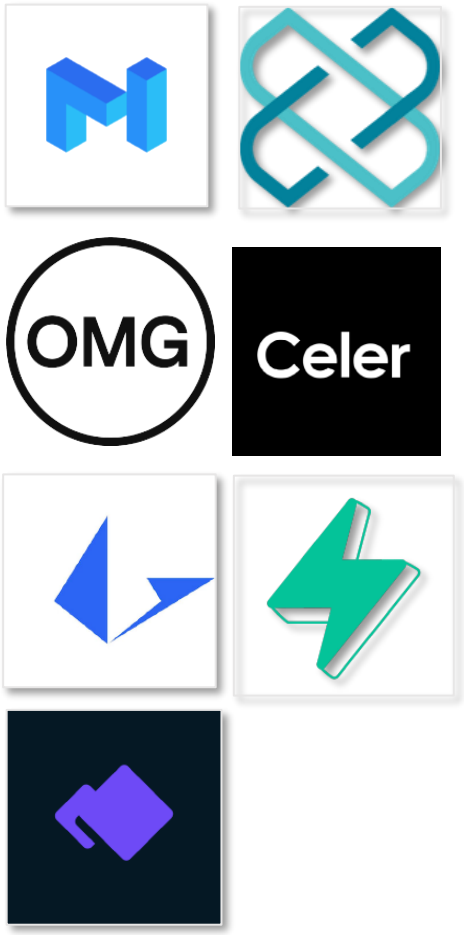
https://medium.com/avalabs/the-avalanche-ethereum-bridge-what-you-need-to-know-b450d2ece03c

https://github.com/poanetwork/token-bridge

https://docs.oceanprotocol.com/architecture/token-bridge

PD-15.3.0 Layer 2

- Optimism
- Matic
- Loom
- Omg (OmiseGO)
- zksync
- Celer
- Arbitrum
- Loopring
- Fuel
- Deversifi



Optimism

zkSync

OFFCHAIN LABS

https://github.com/ethereum-optimism/optimism-monorepo

https://github.com/maticnetwork

https://github.com/loomnetwork

https://github.com/omgnetwork

https://github.com/matter-labs/zksync

https://github.com/celer-network

https://github.com/OffchainLabs/arbitrum

https://github.com/Loopring

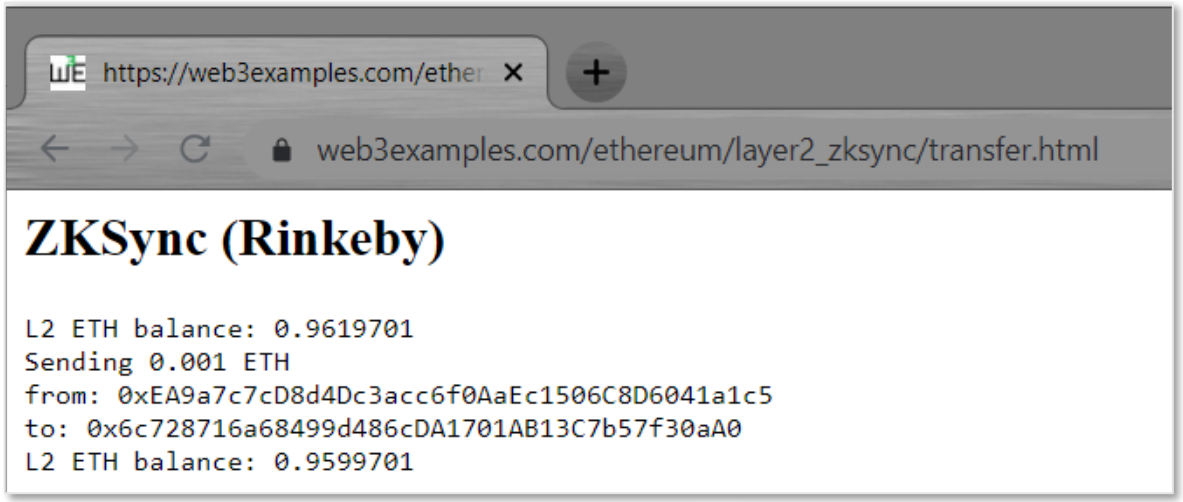
https://github.com/deversifi

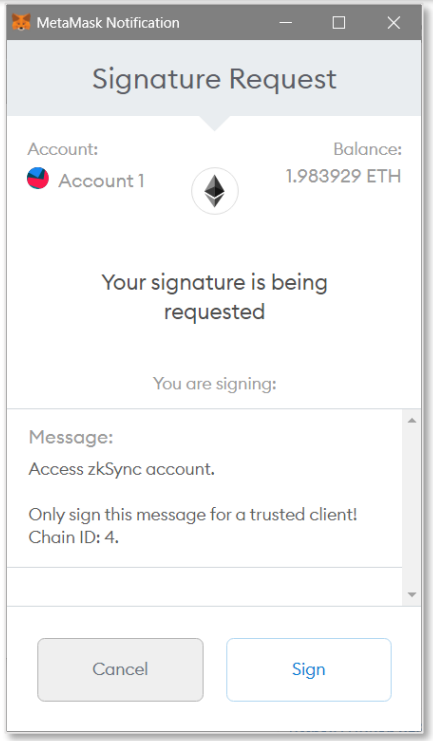
https://medium.com/matter-labs/evaluating-ethereum-l2-scaling-solutions-a-comparison-framework-b6b2f410f955

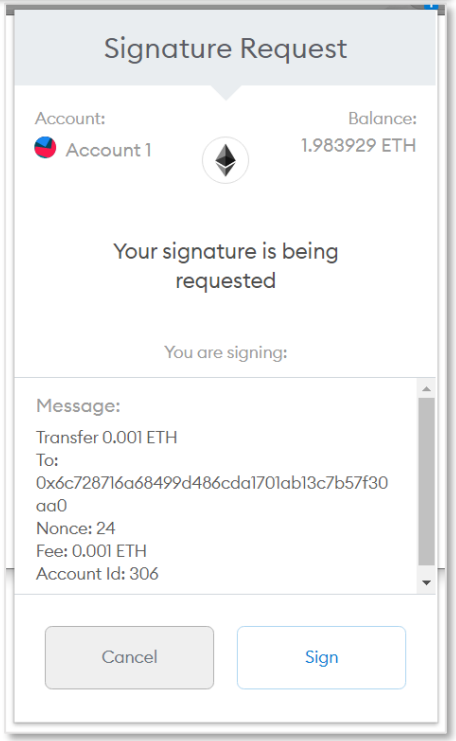
PD-15.3.1 Zero Knowledge terms

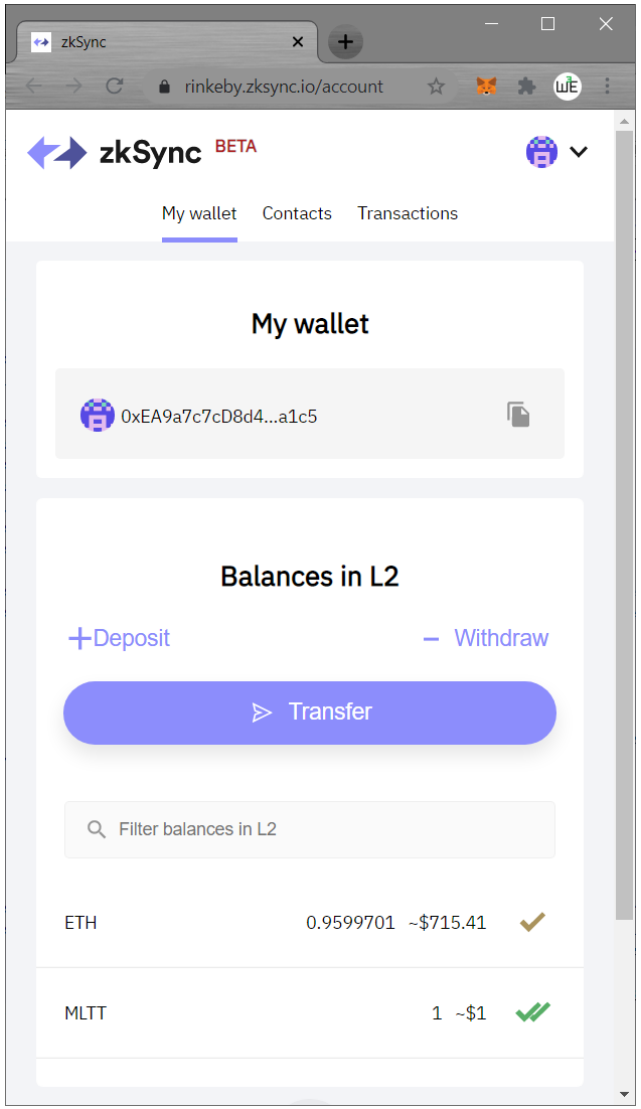
Abbreviations	Meaning
ZK	Zero-Knowledge
Succinct	Short and to the point / verifiable in short time (requires trusted setup)
Non-interactive	One message (so no need for multiple rounds)
SNARK	Succinct Non-interactive adaptive ARgument of Knowledge
Argument	Proof
Transparent	No trusted setup
STARK	Scalable Transparent ARguments of Knowledge (quantum-resistant)
Bulletproef	Short non-interactive zero-knowledge proofs that require no trusted setup (range proofs) (not quantum-resistant)
R1CS	Rank-1 Constraint System

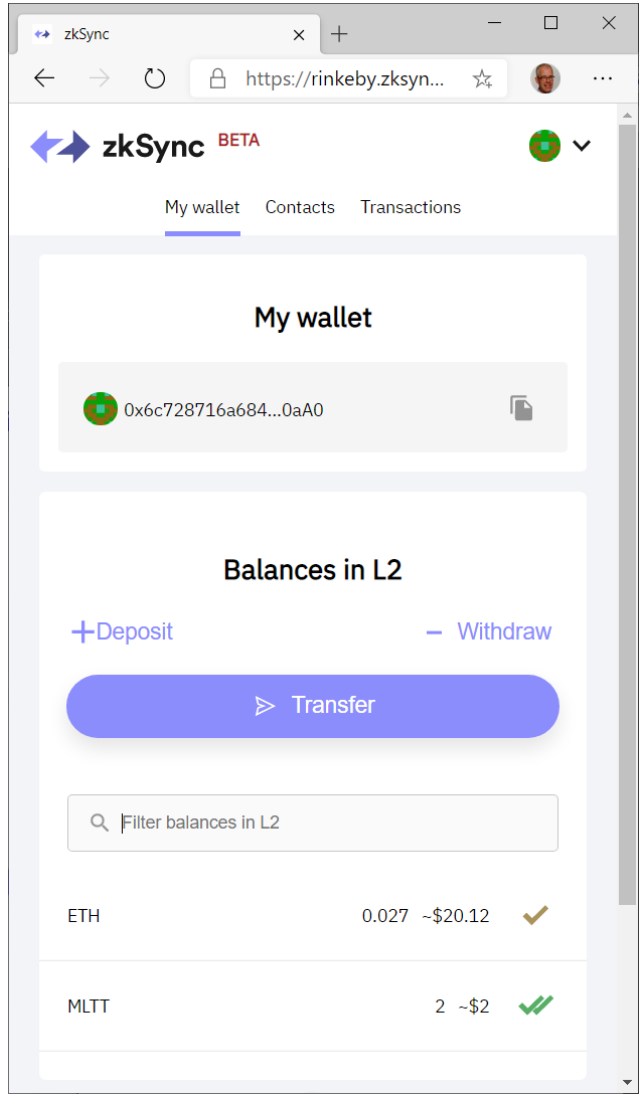
PD-15.3.2 ZKSync











https://rinkeby.zksync.io/account

https://rinkeby.zkscan.io

https://web3examples.com/ethereum/layer2_zksync/transfer.html

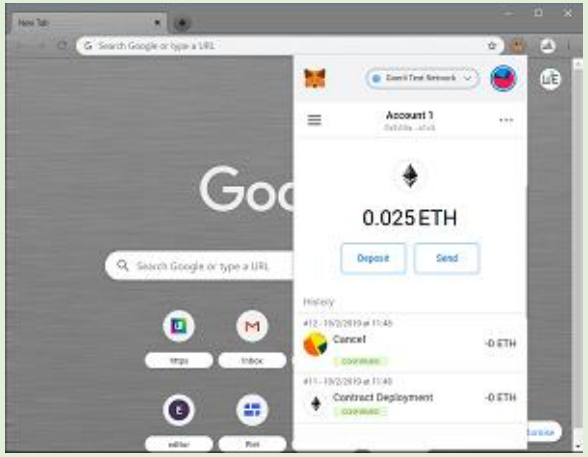
PD-15.3.2 ZKSync

```
await zksync.crypto.loadZkSyncCrypto();
const provider = new ethers.providers.Web3Provider(window.ethereum)
await window.ethereum.enable();
let accounts = await provider.listAccounts()
const signer = provider.getSigner()
const bcnetwork = await provider.getNetwork();
if (bcnetwork.chainId !=4) {log("Select Rinkeby");return; }
const zksProvider = await zksync.getDefaultProvider("rinkeby");
const SyncWallet = await zksync.Wallet.fromEthSigner(signer, zksProvider); // login (by signing a message)
if (!await SyncWallet.isSigningKeySet()) {
    if ((await SyncWallet.getAccountId()) == undefined) { log('Unknown account');return; }
    const changePubkey = await SyncWallet.setSigningKey({ feeToken: 'ETH' }); // requires fee
    const receipt = await changePubkey.awaitReceipt();         // Wait till transaction is committed
}
log(`L2 ETH balance: ${ethers.utils.formatEther(await SyncWallet.getBalance("ETH" ))}`);
var transfer={
    to:     "0x6c728716a68499d486cDA1701AB13C7b57f30aA0",
    token:  "0x0000000000000000000000000000000000000000",//ETH
    amount: ethers.utils.parseEther("0.001"),
    fee:    ethers.utils.parseEther("0.001")
}
log(`Sending ${ethers.utils.formatEther(transfer.amount)} ETH<br>from: ${accounts[0]}<br>to: ${transfer.to}`)
const transferTransaction  = await SyncWallet.syncTransfer(transfer)
const transactionReceipt   = await transferTransaction.awaitReceipt();
log(`L2 ETH balance: ${ethers.utils.formatEther(await SyncWallet.getBalance("ETH" ))}`);
```

https://github.com/web3examples/ethereum/blob/master/layer2_zksync/transfer.html

PD-15.4 MetaMask Snaps

PC : Browser + MetaMask



Snaps

MetaMask Extension

Accounts & Private keys

https://github.com/MetaMask/metamask-snaps-beta/wiki/Snaps-API

Chrome

Web page (html & javascript)

Web3.js

Injected provider

Web3.js interface

Provider Interface

RPC interface

Gateway

Infura

RPC

Ethereum Blockchain

https://web3js.readthedocs.io/en/latest

https://eth.wiki/json-rpc/API

https://eips.ethereum.org/EIPS/eip-1193

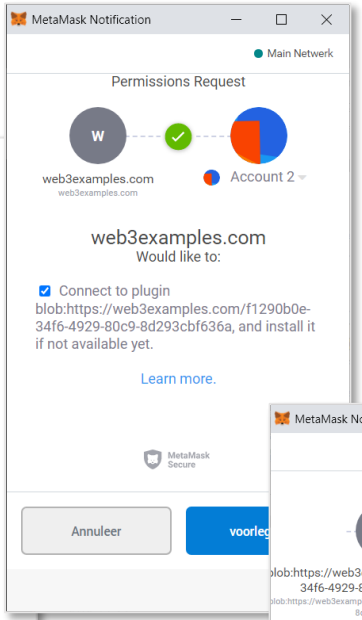
https://metamask.github.io/metamask-docs/

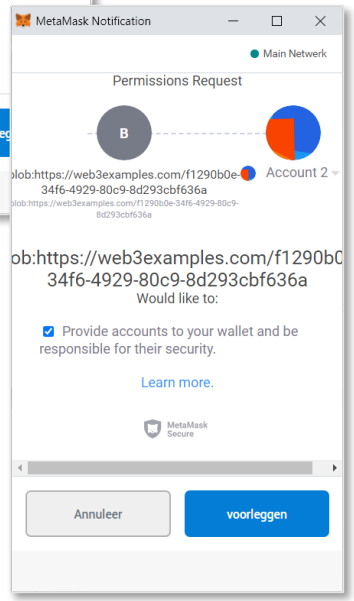
PD-15.4 MetaMask Snaps

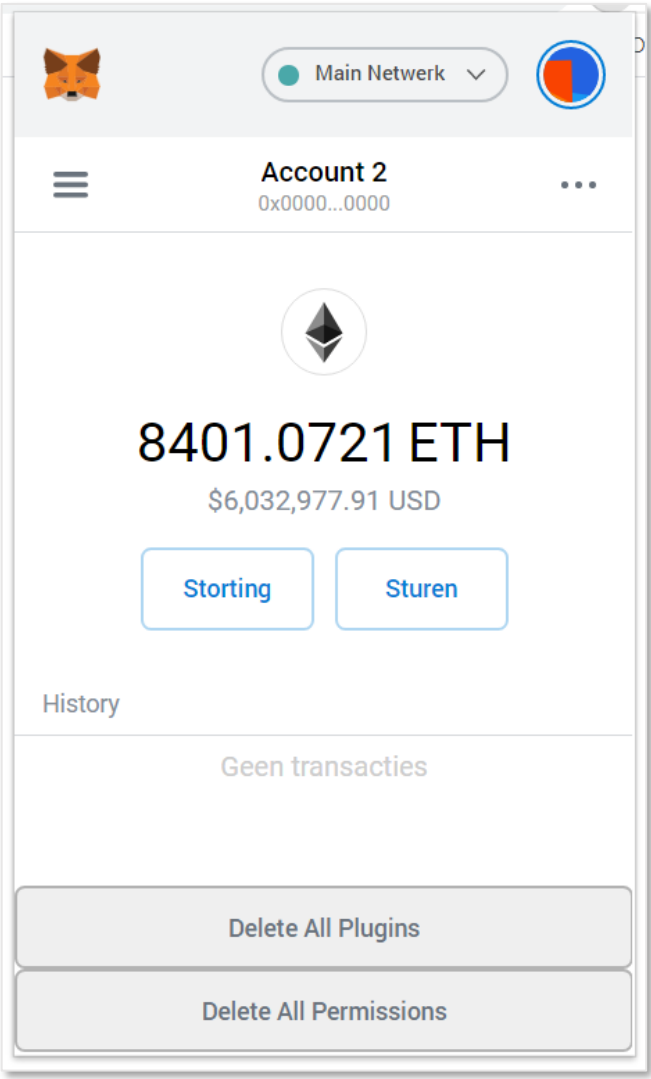
MetaMask snap: add account

Add

```
blob:https://web3examples.com/52b37e3e-69c5-4687-b8cf-b89768840812:
async () => {
            const callparam={
                method: 'wallet_manageIdentities',
                params: [ 'add', { address: "0x0000000000000000000000000000000000000000"}]
            }
            await wallet.send(callparam)
        }
blob:https://web3examples.com/e7035a29-071a-40b9-a66a-a48334e96516:
{
    "web3Wallet": {
        "bundle": {
            "url": "blob:https://web3examples.com/52b37e3e-69c5-4687-b8cf-b89768840812"
        },
        "initialPermissions": {
            "wallet_manageIdentities": {}
        }
    }
}
ethereum.send:
{
    "method": "wallet_enable",
    "params": [
        {
            "wallet_plugin": {
                "blob:https://web3examples.com/e7035a29-071a-40b9-a66a-a48334e96516": {}
            }
        }
    ]
}
```







https://github.com/MetaMask/metamask-snaps-beta/releases

https://github.com/NodeFactoryIo/metamask-snaps-beta/releases

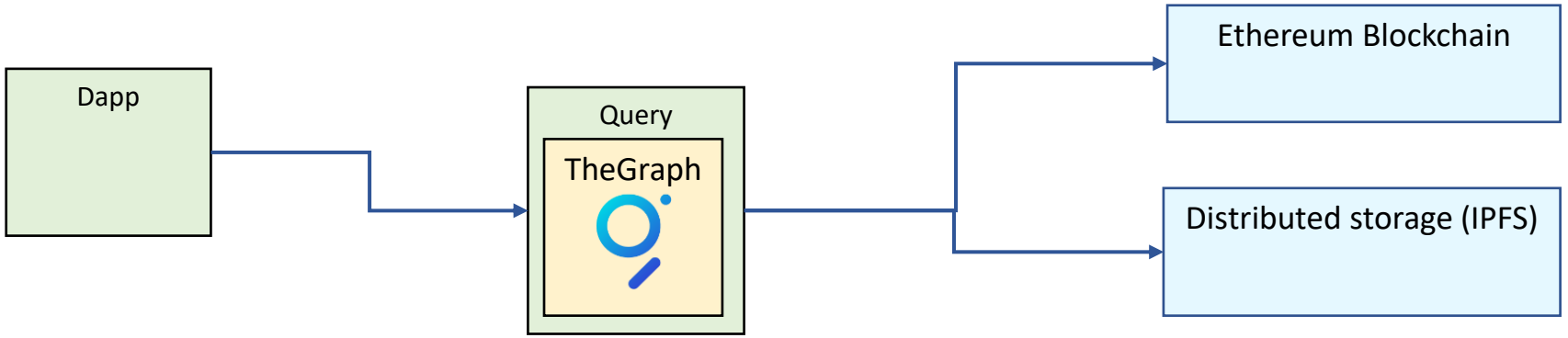
https://web3examples.com/ethereum/snaps_examples/addaccount.html

PD-15.4 MetaMask Snaps

```
var snap= // this is the code of the snap, that will run in the metamask environment
`....
    const callparam={
        method: 'wallet_manageIdentities',
        params: ['add', {address: "0x0000000000000000000000000000000000000000"}]
    }
    await wallet.send(callparam)
`
var wrap=`async () => { ${snap} }` // wrap see https://github.com/MetaMask/snaps-cli
const blobsnap = new Blob([wrap], {type : 'application/javascript'});
var urlsnap=URL.createObjectURL(blobsnap)   // this is a link to the wrapped snap, so it can be loaded by metamask
var package={ // this is the source for the "package.json" that links to the snap and request premissions
    "web3Wallet": {
        "bundle": { "url": urlsnap },// this is the link to the snap
        "initialPermissions": { "wallet_manageIdentities": {} } // these are the permissions
    }
}
var str=JSON.stringify(package, null, 4)
const blob = new Blob([str], {type : 'application/json'});
var url=URL.createObjectURL(blob) // this is a link to the package, which can be loaded by metamask
log(`${urlsnap}:<br>${wrap}`)
log(`${url}:<br>${str}`)
var cmd={ // install snap & get permissions
    method: 'wallet_enable',
    params: [{ wallet_plugin: { [url]: {} } }]
}
log(`ethereum.send:<br>${JSON.stringify(cmd, null, 4)}`)
await ethereum.send(cmd)
```

https://github.com/web3examples/ethereum/blob/master/snaps_examples/addaccount.html

PD-15.5 The Graph

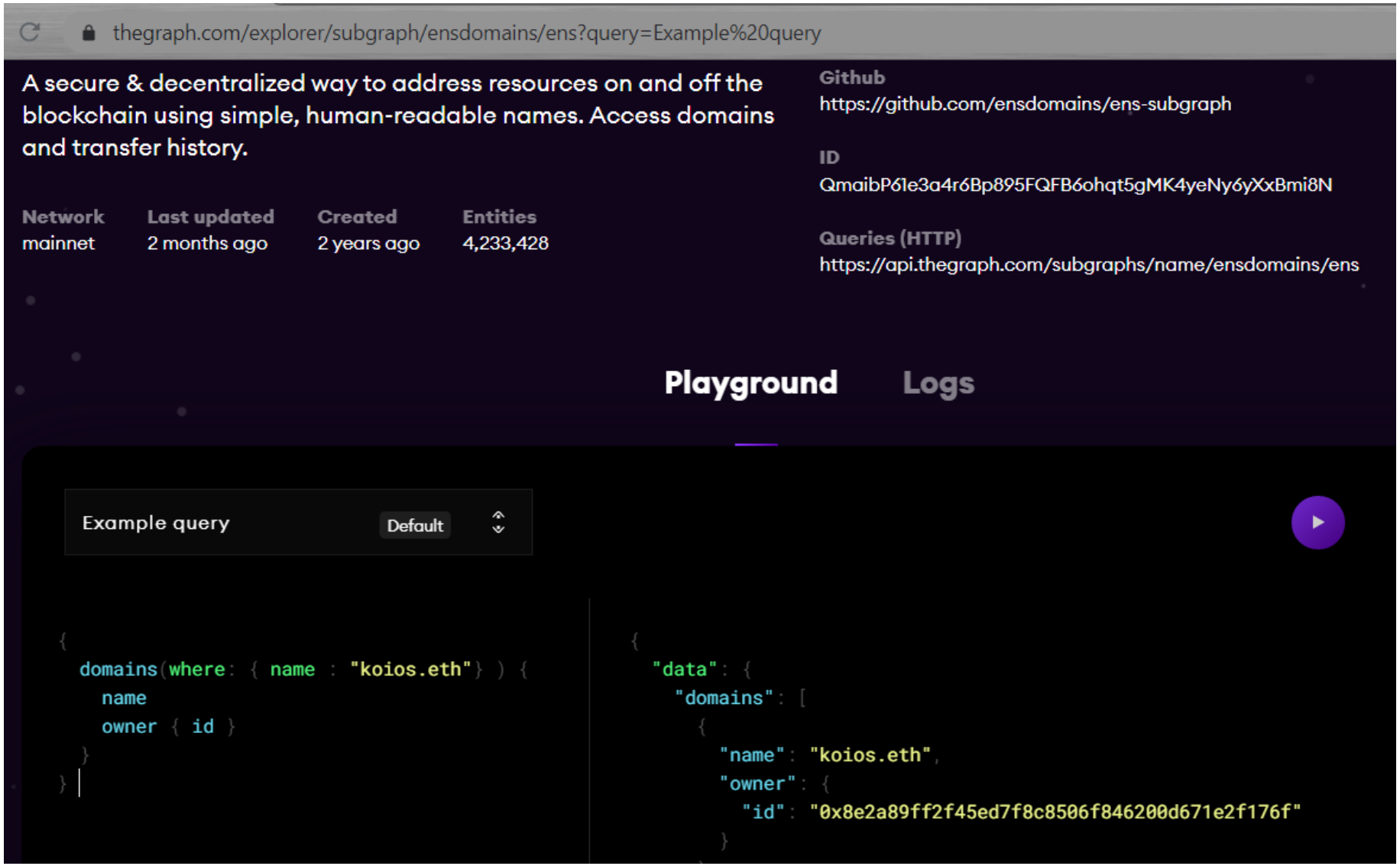


https://thegraph.com

https://thegraph.com/docs/define-a-subgraph

https://ethereumdev.io/how-to-access-indexed-ethereum-data-with-graph

PD-15.5 The Graph Explorer



https://graphql.org

https://thegraph.com/explorer

PD-15.5 The Graph ENS

ens.html

```
<!-- https://thegraph.com/explorer/subgraph/ensdomains/ens
-->
<!DOCTYPE html>
<html>
    <body>
        <h1>ENS Name owner</h1>
        <pre id="log" style="width:100%;height:200px"></pre>
        <script type="text/javascript">
        function log(logstr) {
            document.getElementById("log").innerHTML +=logstr+"\n";
        }
        async function f() {
            const query=`
                {
                    domains(where: { name : "koios.eth"} ) {
                        name
                        owner { id }
                    }
                }
                `
            const URL = 'https://api.thegraph.com/subgraphs/name/ensdomains/ens';
            let body = JSON.stringify({query: query});
            var res=await fetch(URL, {
                method: 'post',
                headers: {'Content-Type': 'application/json'},
                body: body
            })
            var json=await res.json()
            log(JSON.stringify(json))
        }
        f();
        </script>
    </body>
</html>
```

https://web3examples.com/ethereum/thegraph_examples/ens.html

https://github.com/web3examples/ethereum/blob/master/thegraph_examples/ens.html

PD-15.5 The Graph Aave flash loans (html)

flash.html

```
<!DOCTYPE html>
<html>
    <body>
        <h1>Flash</h1>
        <pre id="log" style="width:100%;height:200px"></pre>
        <script type="text/javascript">
        function log(logstr) {
            document.getElementById("log").innerHTML +=logstr+"\n";
        }
        async function f() {
            const query=`
                {
                    flashLoans(first: 10, orderBy: timestamp, orderDirection: desc) {
                        id
                        reserve {
                            name
                            symbol
                        }
                        amount,
                        target,
                        timestamp
                    }
                }
                `
            const URL = 'https://api.thegraph.com/subgraphs/name/aave/protocol';
            let body = JSON.stringify({query: query});
            var res=await fetch(URL, {
                method: 'post',
                headers: {'Content-Type': 'application/json'},
                body: body
            })
            var json=await res.json()
            for (const flashsloan of json.data.flashLoans)
                log(JSON.stringify(flashsloan) )
        }
        f();
        </script>
    </body>
</html>
```

https://web3examples.com/ethereum/thegraph_examples/flash.html

https://github.com/web3examples/ethereum/blob/master/thegraph_examples/flash.html