PD-15 New Developments

PD-15.1 MetaTransactions

c) Meta Transaction flow
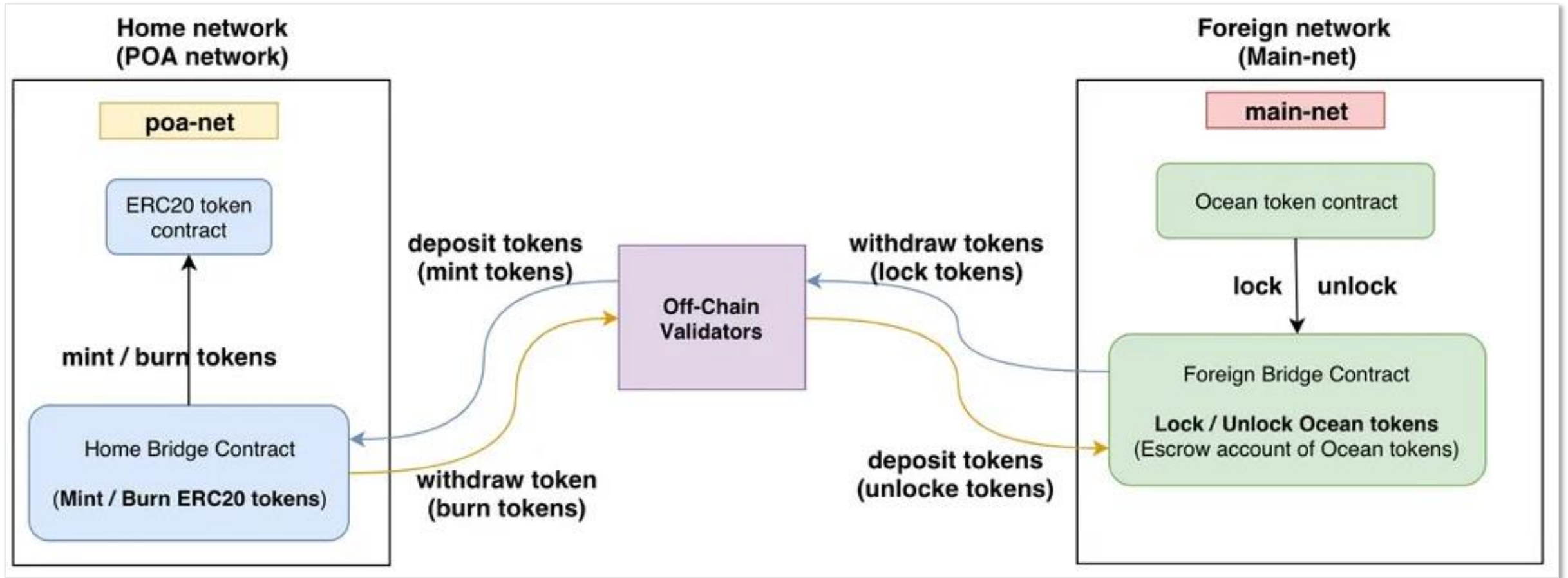
2. Sign transaction: including bytes for App Contract

3. Send user signature to Relay Contract

4. Call App Contract with byte data from user

User

App Server

Relay Contract

App Contract

1. Serve front-end app files

4. User watch for event to prove action occurred correctly.

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
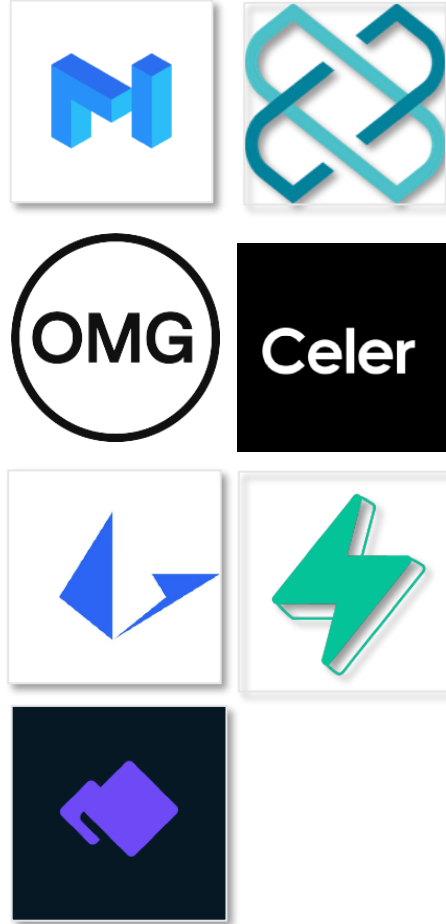
PD-15.2 Bridges

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

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

https://github.com/deversifi

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)
Bulletproof	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://web3examples.com/ethereum/layer2_zksync/transfer.html

ZKSync (Rinkeby)

```
L2 ETH balance: 0.9619701
Sending 0.001 ETH
from: 0xEA9a7c7cD8d4Dc3acc6f0AaEc1506C8D6041a1c5
to: 0x6c728716a68499d486cDA1701AB13C7b57f30aA0
L2 ETH balance: 0.9599701
```

MetaMask Notification

Signature Request

Account:
Account 1

Balance:
1.983929 ETH

Your signature is being requested

You are signing:

Message:

Access zkSync account.

Only sign this message for a trusted client!
Chain ID: 4.

Cancel   Sign

Signature Request

Account:
Account 1

Balance:
1.983929 ETH

Your signature is being requested

You are signing:

Message:

Transfer 0.001 ETH
To:
0x6c728716a68499d486cda1701ab13c7b57f30
aa0
Nonce: 24
Fee: 0.001 ETH
Account Id: 306

Cancel   Sign

---

zkSync

rinkeby.zksync.io/account

zkSync BETA

My wallet   Contacts   Transactions

My wallet

0xEA9a7c7cD8d4...a1c5

Balances in L2

+ Deposit   — Withdraw

▷ Transfer

Filter balances in L2

ETH   0.9599701 ~$715.41 ✓

MLTT   1 ~$1 ✓✓

---

zkSync

https://rinkeby.zksync...

zkSync BETA

My wallet   Contacts   Transactions

My wallet

0x6c728716a684...0aA0

Balances in L2

+ Deposit   — Withdraw

▷ Transfer

Filter balances in L2
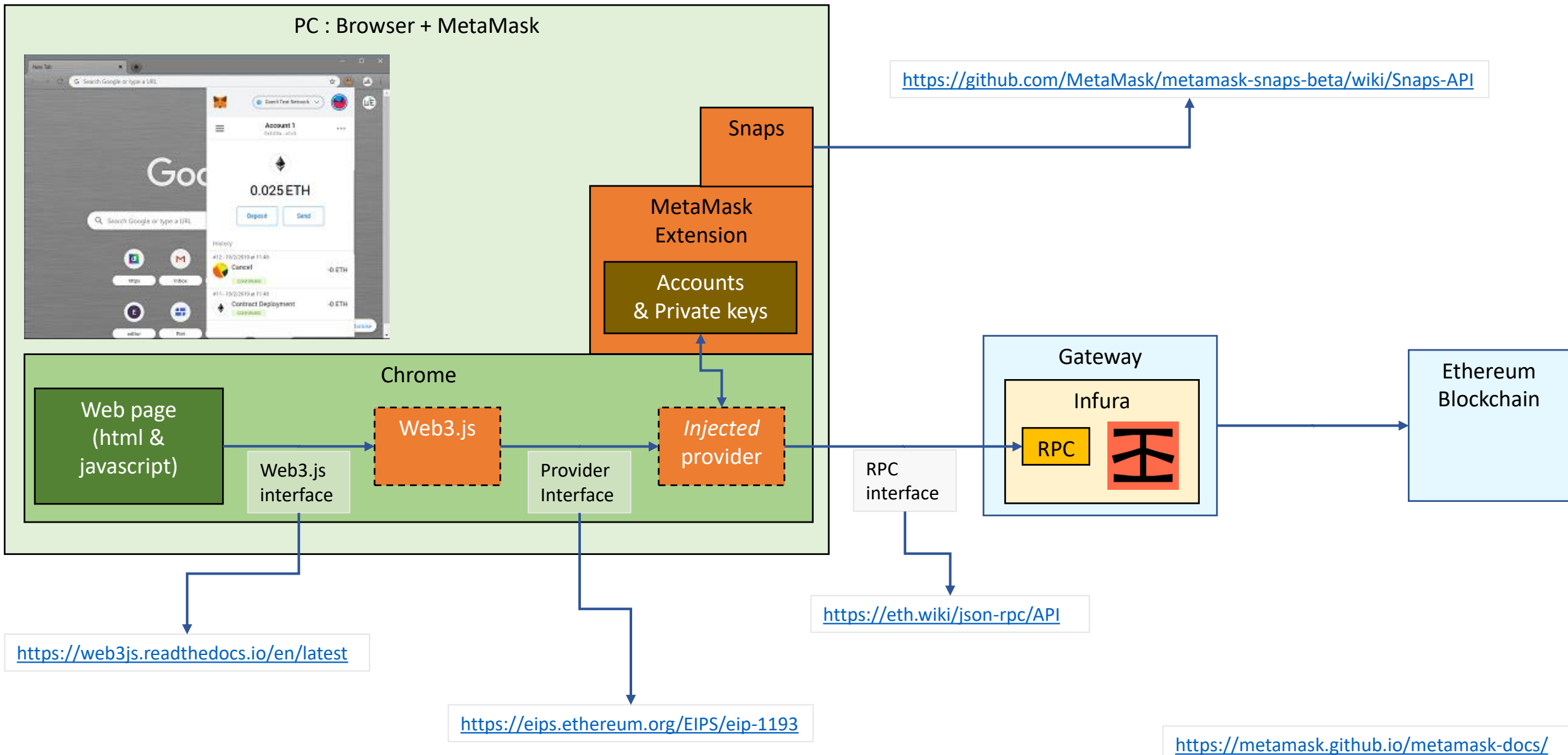
ETH   0.027 ~$20.12 ✓

MLTT   2 ~$2 ✓✓

https://rinkeby.zksync.io/account

https://rinkeby.zkscan.io

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

PD-15.3.2 ZKSync

```javascript
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"))}`);
```

PD-15.4 MetaMask Snaps

PC : Browser + MetaMask
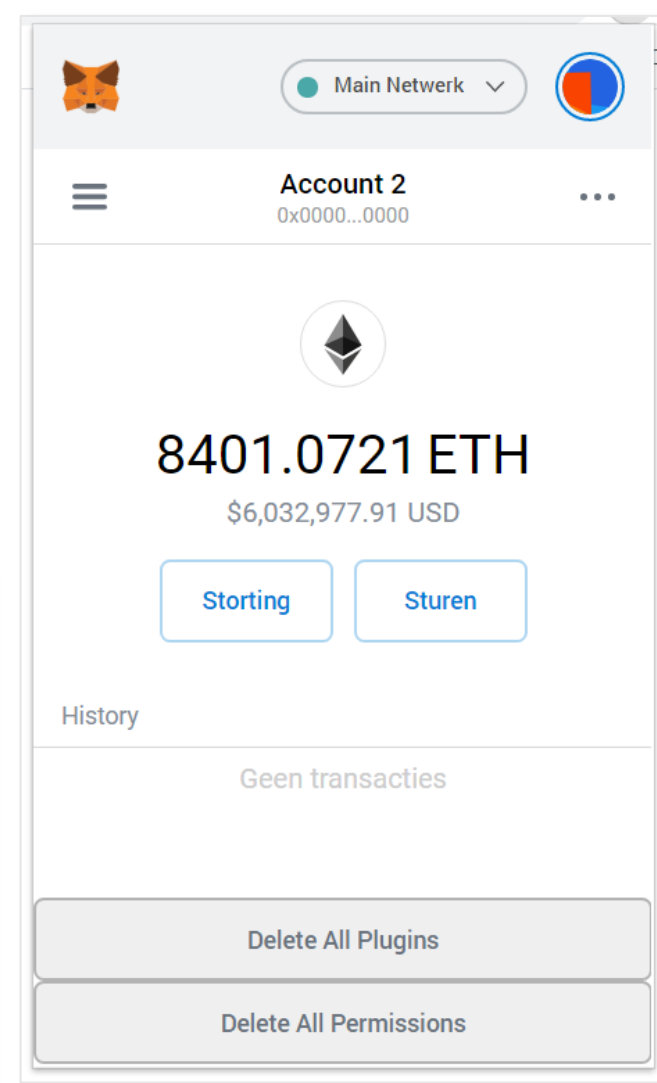
Snaps

MetaMask
Extension

Accounts
& Private keys

Chrome

Web page
(html &
javascript)

Web3.js

Injected
provider

Web3.js
interface

Provider
Interface

RPC
interface

Gateway

Infura

RPC

**Ethereum
Blockchain**

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

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

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

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

https://metamask.github.io/metamask-docs/
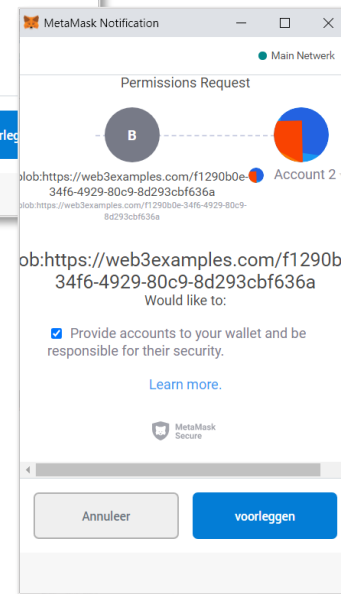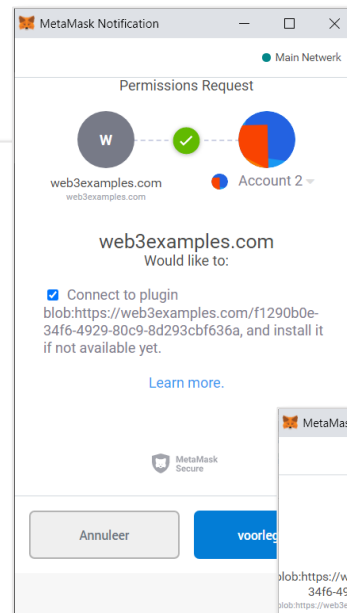
PD-15.4 MetaMask Snaps

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": {}
            }
        }
    ]
}
```
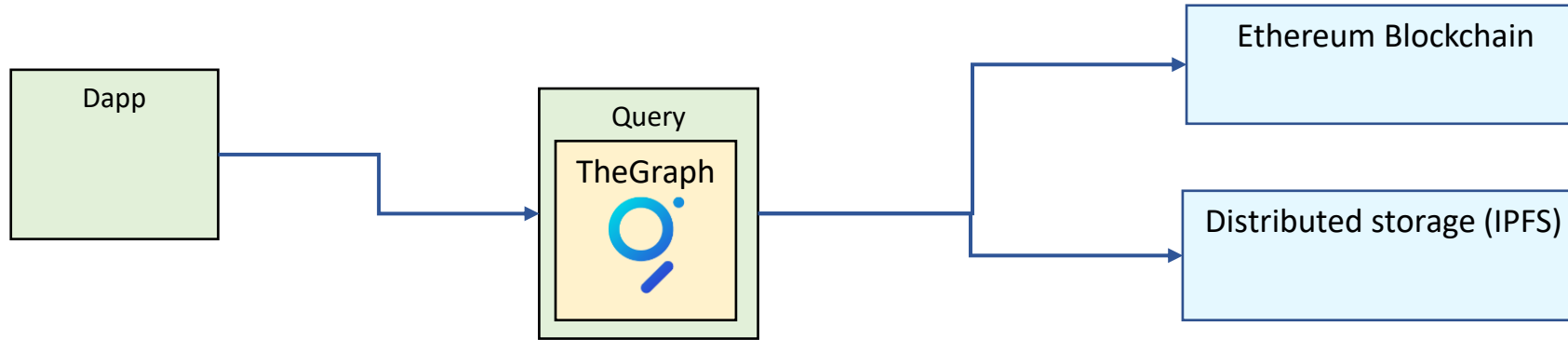
PD-15.4 MetaMask Snaps

```javascript
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

Dapp → Query — TheGraph → Ethereum Blockchain / Distributed storage (IPFS)

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

PD-15.5 The Graph ENS

```html
ens.html
1  <!-- https://thegraph.com/explorer/subgraph/ensdomains/ens
2  -->
3  <!DOCTYPE html>
4  <html>
5      <body>
6          <h1>ENS Name owner</h1>
7          <pre id="log" style="width:100%;height:200px"></pre>
8          <script type="text/javascript">
9          function log(logstr) {
10             document.getElementById("log").innerHTML +=logstr+"\n";
11         }
12         async function f() {
13             const query=`
14                 {
15                     domains(where: { name : "koios.eth"} ) {
16                         name
17                         owner { id }
18                     }
19                 }
20                 `
21             const URL = 'https://api.thegraph.com/subgraphs/name/ensdomains/ens';
22             let body = JSON.stringify({query: query});
23             var res=await fetch(URL, {
24                 method: 'post',
25                 headers: {'Content-Type': 'application/json'},
26                 body: body
27             })
28             var json=await res.json()
29             log(JSON.stringify(json) )
30         }
31         f();
32         </script>
33     </body>
34 </html>
```

PD-15.5 The Graph Aave flash loans (html)

```html
flash.html

1    <!DOCTYPE html>
2    <html>
3        <body>
4            <h1>Flash</h1>
5            <pre id="log" style="width:100%;height:200px"></pre>
6            <script type="text/javascript">
7            function log(logstr) {
8                document.getElementById("log").innerHTML +=logstr+"\n";
9            }
10           async function f() {
11               const query=`
12                   {
13                       flashLoans(first: 10, orderBy: timestamp, orderDirection: desc) {
14                           id
15                           reserve {
16                               name
17                               symbol
18                           }
19                           amount,
20                           target,
21                           timestamp
22                       }
23                   }
24               `
25               const URL = 'https://api.thegraph.com/subgraphs/name/aave/protocol';
26               let body = JSON.stringify({query: query});
27               var res=await fetch(URL, {
28                   method: 'post',
29                   headers: {'Content-Type': 'application/json'},
30                   body: body
31               })
32               var json=await res.json()
33               for (const flashsloan of json.data.flashLoans)
34                   log(JSON.stringify(flashsloan) )
35           }
36           f();
37           </script>
38       </body>
39   </html>
```