Metamask

Contents

What is Metamask

Docs: https://docs.metamask.io/guide/#why-metamask

MetaMask allows users to manage accounts and their keys in a variety of ways, including hardware wallets. Whenever you request a transaction signature (like eth_sendTransaction), MetaMask will prompt the user. MetaMask comes pre-loaded with fast connections to the Ethereum blockchain and several test networks via Infura. This allows you to get started without synchronizing a full node. MetaMask is compatible with any blockchain that exposes an Ethereum-compatible JSON RPC API.

Quick Start

Once MetaMask is installed and running, you should find that new browser tabs have a window.ethereum object available in the developer console. This is how your website will interact with MetaMask.

Resetting Your Local Nonce Calculation:

If you're running a test blockchain and restart it, you can accidentally confuse MetaMask because it calculates the next nonce based on both the network state and the known sent transactions. To clear MetaMask's transaction queue, and effectively reset its nonce calculation, you can use the Reset Account button in Settings.

Get current network and address: ethereum.networkVersion and ethereum.selectedAddress

Important methods/events:

Enable a DAPP - select account(s) (via popup):

```
const accounts = await ethereum.request({ method: 'eth_requestAccounts' });
```

Check the current account: ethereum.selectedAddress

```
ethereum.on('accountsChanged', function (accounts) {
  // Time to reload your interface with accounts[0]!
});
```

Sending a transaction:

```
const transactionParameters = {
  nonce: '0x00', // ignored by MetaMask
  gasPrice: '0x09184e72a000', // customizable by user during MetaMask confirmation.
  gas: '0x2710', // customizable by user during MetaMask confirmation.
```

```
  to: '0x0000000000000000000000000000000000000000', // Required except during contract publications.
  from: ethereum.selectedAddress, // must match user's active address.
  value: '0x00', // Only required to send ether
  data: '0x7f7465737432000000000000000000...',
  chainId: '0x3', // Used to prevent transaction reuse across blockchains. Auto-filled by MetaMask.
};

const txHash = await ethereum.request({
  method: 'eth_sendTransaction',
  params: [transactionParameters],
});
```

Ethereum Provider API

MetaMask injects a global API into websites at window.ethereum. This API allows websites to request users' Ethereum accounts, read data from blockchains the user is connected to, and suggest that the user sign messages and transactions. Most developers use a convenience library, such as ethers, instead of using the provider directly.

For any Ethereum web application you will have to:

Detect the Ethereum provider (window.ethereum)
Detect which Ethereum network the user is connected to
Get the user's Ethereum account(s)

Properties:

ethereum.isMetaMask

Methods:

Use request to submit RPC requests to Ethereum via MetaMask: ***ethereum.request(args***)

```
params: [
  {
    from: '0xb60e8dd61c5d32be8058bb8eb970870f07233155',
    to: '0xd46e8dd67c5d32be8058bb8eb970870f07244567',
    value: '0x9184e72a',
  },
];
```

```
const result = ethereum.request({
   method: 'eth_sendTransaction',
   params,
 })
```

Events:

accountsChanged:

```
ethereum.on('accountsChanged', (accounts) => {
  // "accounts" will always be an array, but it can be empty.
});
```

chainChanged:

```
ethereum.on('chainChanged', (chainId) => {
  // We recommend reloading the page
  window.location.reload();
});
```

Remove listeners once you are done listening to them:

```
ethereum.on('accountsChanged', handleAccountsChanged);
ethereum.removeListener('accountsChanged', handleAccountsChanged);
```

disconnect:

The MetaMask provider emits this event if it becomes unable to submit RPC requests. You can also use the ethereum.isConnected() method to determine if the provider is disconnected

```
ethereum.on('disconnect', handler: (error: ProviderRpcError) => void);
```

Errors

All errors thrown or returned by the MetaMask provider follow this interface:

```
interface ProviderRpcError extends Error {
  message: string;
  code: number;
  data?: unknown;
}
```

Example

This snippet explains how to accomplish the three most common requirements for web3 sites:

```
import detectEthereumProvider from '@metamask/detect-provider';

// this returns the provider, or null if it wasn't detected
const provider = await detectEthereumProvider();

if (provider) {
  startApp(provider); // Initialize your app
} else {
  console.log('Please install MetaMask!');
}

function startApp(provider) {
  // If the provider returned by detectEthereumProvider is not the same as
  // window.ethereum, something is overwriting it, perhaps another wallet.
  if (provider !== window.ethereum) {
    console.error('Do you have multiple wallets installed?');
  }
  // Access the decentralized web!
}

/**********************************************************/
/* Handle chain (network) and chainChanged (per EIP-1193) */
/**********************************************************/

const chainId = await ethereum.request({ method: 'eth_chainId' });
handleChainChanged(chainId);

ethereum.on('chainChanged', handleChainChanged);

function handleChainChanged(_chainId) {
  // We recommend reloading the page, unless you must do otherwise
  window.location.reload();
}

/***********************************************************/
/* Handle user accounts and accountsChanged (per EIP-1193) */
/***********************************************************/

let currentAccount = null;
ethereum
  .request({ method: 'eth_accounts' })
  .then(handleAccountsChanged)
```

```
  .catch((err) => {
    // Some unexpected error.
    // For backwards compatibility reasons, if no accounts are available,
    // eth_accounts will return an empty array.
    console.error(err);
  });

// Note that this event is emitted on page load.
// If the array of accounts is non-empty, you're already
// connected.
ethereum.on('accountsChanged', handleAccountsChanged);

// For now, 'eth_accounts' will continue to always return an array
function handleAccountsChanged(accounts) {
  if (accounts.length === 0) {
    // MetaMask is locked or the user has not connected any accounts
    console.log('Please connect to MetaMask.');
  } else if (accounts[0] !== currentAccount) {
    currentAccount = accounts[0];
    // Do any other work!
  }
}

/*********************************************/
/* Access the user's accounts (per EIP-1102) */
/*********************************************/

// You should only attempt to request the user's accounts in response to user
// interaction, such as a button click.
// Otherwise, you popup-spam the user like it's 1999.
// If you fail to retrieve the user's account(s), you should encourage the
//user to initiate the attempt.
document.getElementById('connectButton', connect);

// While you are awaiting the call to eth_requestAccounts, you should disable
// any buttons the user can click to initiate the request.
// MetaMask will reject any additional requests while the first is still
// pending.
function connect() {
  ethereum
    .request({ method: 'eth_requestAccounts' })
    .then(handleAccountsChanged)
    .catch((err) => {
      if (err.code === 4001) {
        // EIP-1193 userRejectedRequest error
        // If this happens, the user rejected the connection request.
        console.log('Please connect to MetaMask.');
      } else {
        console.error(err);
```

```
      }
    });
}
```

RPC API

MetaMask uses the ethereum.request(args) method to wrap an RPC API.

Ethereum JSON-RPC Methods

https://ethereum.org/en/developers/docs/apis/json-rpc/#json-rpc-methods

eth_accounts(opens new window)
eth_call(opens new window)
eth_getBalance(opens new window)
eth_sendTransaction(opens new window)
eth_sign(opens new window)

Restricted Methods

MetaMask introduced Web3 Wallet Permissions via EIP-2255 (opens new window). In this permissions system, each RPC method is either restricted or unrestricted. If a method is restricted, the caller must have the corresponding permission in order to call it.

`eth_requestAccounts`

Requests that the user provides an Ethereum address to be identified by. The request causes a MetaMask popup to appear. You should only request the user's accounts in response to user action, such as a button click.

```
ethereum
    .request({ method: 'eth_requestAccounts' })
    .then(handleAccountsChanged)
    .catch((error) => {
      if (error.code === 4001) {
        // EIP-1193 userRejectedRequest error
        console.log('Please connect to MetaMask.');
      } else {
        console.error(error);
      }
    });
```

`wallet_switchEthereumChain`

Creates a confirmation asking the user to switch to the chain with the specified chainId.

As with any method that causes a confirmation to appear, wallet_switchEthereumChain should only be called as a result of direct user action, such as the click of a button.

```
try {
  await ethereum.request({
    method: 'wallet_switchEthereumChain',
    params: [{ chainId: '0xf00' }],
  });
} catch (switchError) {
  // This error code indicates that the chain has not been added to MetaMask.
  if (switchError.code === 4902) {
    try {
      await ethereum.request({
        method: 'wallet_addEthereumChain',
        params: [
          {
            chainId: '0xf00',
            chainName: '...',
            rpcUrls: ['https://...'] /* ... */,
          },
        ],
      });
    } catch (addError) {
      // handle "add" error
    }
```

Signing Data

The method signTypedData_v3 currently represents the latest version of the ***EIP-712*** spec (opens new window), making it the most secure method for signing cheap-to-verify data on-chain that we have yet.

Example: https://medium.com/metamask/eip712-is-coming-what-to-expect-and-how-to-use-it-bb92fd1a7a26