# Retrieve Root Chain and Child Chain Balances

Retrieving balances involves converting an RLP encoded array of balances into a human-readable array of balances.

### **Implementation**

1. Install <a href="mailto:omg-js">omg-js</a>, <a href="web3">web3</a>

To access network features from your application, use our official libraries:

Node Browser React Native

Requires Node >= 8.11.3 < 13.0.0

```
npm install @omisego/omg-js web3

JavaScript (ESNext)
```

#### 2. Import dependencies

omg-js library has 3 main objects that are used during all of the code samples. Here's an example of how to instantiate them:

```
import Web3 from "web3";
import { ChildChain, RootChain, OmgUtil } from "@omisego/omg-js";

// instantiate omg-js objects
const web3 = new Web3(new Web3.providers.HttpProvider(web3_provider_url));
const rootChain = new RootChain({ web3, plasmaContractAddress });
const childChain = new ChildChain({ watcherUrl });

// define constants
const erc20ContractAddress = "0xd92e713d051c37ebb2561803a3b5fbabc4962431";
const aliceAddress = "0x8cb0de6206f459812525f2ba043b14155c2230c0";
```

- web3\_provider\_url the URL to a full Ethereum RPC node (local or from infrastructure provider, e.g. <u>Infura</u>).
- plasmaContractAddress CONTRACT\_ADDRESS\_PLASMA\_FRAMEWORK for defined environment.
- watcherUrl the Watcher Info URL for defined environment (personal or from OMG Network).

#### 3. Retrieve balances

There's no direct way to retrieve balances on both Ethereum and OMG networks. Instead, you first retrieve an <u>RLP encoded</u> array of <u>BigNum</u> balances and then convert it to a preferred format.

#### 3.1 Retrieve child chain (OMG Network) balances

Note, the amount for ERC-20 tokens will be displayed in the lowest denomination of that particular token. You can convert it into a number via <a href="web3.utils.fromwei">web3.utils.fromwei</a>. For example, the provided address has multiple tokens with different decimals (18, 0, 18, 18, 6):

```
    Copy

  "currency": "ETH",
  "amount": "0.29996999999999993"
},
  "currency": "0x2d453e2a14a00f4a26714a82abfc235c2b2094d5",
  "amount": "100"
},
  "currency": "0x5592ec0cfb4dbc12d3ab100b257153436a1f0fea",
  "amount": "1000000000000000000000"
},
  "currency": "0x942f123b3587ede66193aa52cf2bf9264c564f87",
  "amount": "1000000000000000000000"
},
  "currency": "0xd92e713d051c37ebb2561803a3b5fbabc4962431",
  "amount": "687000000"
}
```

For example, if you want to convert the amount for TUSDT token (<u>0xd92e713d051c37ebb2561803a3b5fbabc4962431</u>), you should either create a custom converter or use the web3.utils as follows:

```
& Copyweb3.utils.fromWei(String(i.amount), "mwei")
```

You can find the number of decimals for a given token on one of the blockchain explorers, such as Etherscan.

#### 3.2 Retrieve root chain (Ethereum) balances

```
🔁 Copy
async function retrieveRootChainErc20Balance() {
  const rootchainBalance = await web3.eth.getBalance(aliceAddress);
  const rootchainBalances = [
      currency: "ETH",
      amount: web3.utils.fromWei(String(rootchainBalance), "ether"),
    },
  1;
  const rootchainERC20Balance = await OmgUtil.getErc20Balance({
    web3,
    address: aliceAddress,
    erc20Address: erc20ContractAddress,
  });
  rootchainBalances.push({
    currency: erc20ContractAddress,
    amount: web3.utils.toBN(rootchainERC20Balance).toString(),
  });
```

Note, you can return the ERC20 balance only for one token at a time.

## Lifecycle

- 1. A user calls the getBalance or getErc20Balance function to create a RLP encoded array of balances that contain <u>BigNum</u> objects.
- 2. A user filters an array of balances and returns an array for the desired currency (ETH\_CURRENCY for ETH or ERC20\_CONTRACT\_ADDRESS for ERC20 tokens).
- 3. A user converts the amount of each balance from WEI into a decimal number (optional).

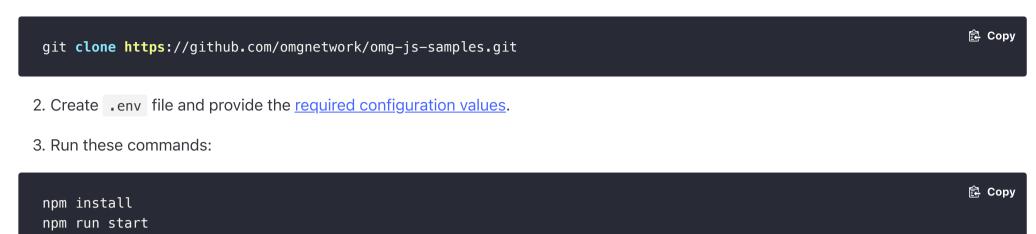
# **Demo Project**

This section provides a demo project that contains a detailed implementation of the tutorial. If you consider integrating with the OMG Network, you can use this sample to significantly reduce the time of development. It also provides step-by-step instructions and sufficient code guidance that is not covered on this page.

### **JavaScript**

For running a full omg-js code sample for the tutorial, please use the following steps:

1. Clone <u>omg-js-samples</u> repository:



- 4. Open your browser at <a href="http://localhost:3000">http://localhost:3000</a>.
- 5. Select Retrieve Balances on the left side, observe the logs on the right.

Code samples for all tutorials use the same repository — omg-js-samples , thus you have to set up the project and install dependencies only one time.