Deposit Funds

A deposit involves sending ETH or ERC-20 tokens to the `Vault` smart contract on the root chain for subsequent use on the OMG network.

Implementation

1. Install [`omg-js`], [`web3`], [`bn.js`]

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

Node   Browser   React Native

Requires Node >= 8.11.3 < 13.0.0

Copy
```
npm install @omisego/omg-js web3 bn.js
```

JavaScript (ESNext)

2. Import dependencies, define constants

Depositing funds to the OMG Network involves using 2 `omg-js` objects. Here's an example of how to instantiate them:

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

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

// define constants
// amount => 0.1 with 18 decimals (ETH) = 100000000000000000
const ethDeposit = {
  amount: new BigNumber("100000000000000000"),
  address: "0x8CB0DE6206f459812525F2BA043b14155C2230C0",
  privateKey: OmgUtil.hexPrefix("CD55F2A7C476306B27315C7986BC50BD81DB4130D4B5CFD49E3EAF9ED1EDE4F7")
}

// amount => 50 with 6 decimals (TUSDT) = 50000000
const erc20Deposit = {
  amount: new BigNumber("50000000"),
  currency: OmgUtil.hexPrefix("0xd92e713d051c37ebb2561803a3b5fbabc4962431"),
  address: "0x8CB0DE6206f459812525F2BA043b14155C2230C0",
  privateKey: OmgUtil.hexPrefix("CD55F2A7C476306B27315C7986BC50BD81DB4130D4B5CFD49E3EAF9ED1EDE4F7")
}
```

- `web3_provider_url` - the URL to a full Ethereum RPC node (local or from infrastructure provider, e.g. Infura).
- `plasmaContractAddress` - `CONTRACT_ADDRESS_PLASMA_FRAMEWORK` for defined environment.

3. Make an ETH deposit

Performing any operation on the OMG Network requires funds. Funds deposit happens when a user sends ETH or ERC20 tokens to the `Vault` smart contract on Ethereum Network. A vault holds custody of tokens transferred to the Plasma Framework. Deposits increase the pool of funds held by the contract and also signals to the child chain server that the funds should be accessible on the child chain.

Copy

```
async function makeEthDeposit () {
  const deposit = await rootChain.deposit({
    amount: ethDeposit.amount,
    currency: OmgUtil.transaction.ETH_CURRENCY,
    txOptions: {
      from: ethDeposit.address,
      privateKey: ethDeposit.privateKey,
      gas: gasLimit
    }
  });
  return deposit;
}
```

- `gasLimit` - gas limit for your transaction. Please check the current data on Gas Station or similar resources.

Deposit amount is defined in WEI, the smallest denomination of ether (ETH), the currency used on the Ethereum network. You can use ETH converter or an alternative tool to know how much WEI you have to put as the `amount` value.

A deposit generates a transaction receipt verifiable on Ethereum Network. A typical receipt has the following structure:

Copy

```
{
    "blockHash": "0x41455ed19db8e5a495233e54c1813962edaf8a5fb87f847a704c72efa90e2c71",
    "blockNumber": 7779244,
    "contractAddress": null,
    "cumulativeGasUsed": 391297,
    "from": "0x0dc8e240d90f3b0d511b6447543b28ea2471401a",
    "gasUsed": 130821,
    "logs": [
        {
            "address": "0x895Cc6F20D386f5C0deae08B08cCFeC9f821E7D9",
            "topics": [
                "0x18569122d84f30025bb8dffb33563f1bdbfb9637f21552b11b8305686e9cb307",
                "0x000000000000000000000000dc8e240d90f3b0d511b6447543b28ea2471401a",
                "0x00000000000000000000000000000000000000000000000000000000000023e42",
                "0x0000000000000000000000000000000000000000000000000000000000000000"
            ],
            "data": "0x000000000000000000000000000000000000000000000000006a94d74f430000",
            "blockNumber": 7779244,
            "transactionHash": "0x0e7d060a63cb65f629cc6d053e71397c7fa3250b41e36cb2cae40b2acb4350a2",
            "transactionIndex": 12,
            "blockHash": "0x41455ed19db8e5a495233e54c1813962edaf8a5fb87f847a704c72efa90e2c71",
            "logIndex": 1,
            "removed": false,
            "id": "log_8b0a6416"
        }
    ],
    "logsBloom": "0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
    "status": true,
    "to": "0x895cc6f20d386f5c0deae08b08ccfec9f821e7d9",
    "transactionHash": "0x0e7d060a63cb65f629cc6d053e71397c7fa3250b41e36cb2cae40b2acb4350a2",
    "transactionIndex": 12
}
```

After the funds are confirmed on the rootchain, the child chain server generates a transaction in a form of UTXO corresponding to the deposited amount. UTXO (unspent transaction output) is a model used to keep a track of balances on the OMG Network.

If a transaction is successful, you will see a unique `transactionHash` that can be verified on Ethereum block explorer, such as Etherscan. Copy the hash and paste it in the search box for the transaction's details.

Depositing also involves forming a pseudo-block on the child chain. Such a block contains a single transaction with the deposited funds as a new UTXO. You can check a new block on the OMG Block Explorer.

4. Make an ERC20 deposit

Depositing ERC20 tokens requires approval of the corresponding `Vault` contract. You can deposit tokens only after this process is finished.

```
async function makeErc20Deposit () {
  // approve ERC20 token
  const approval = await rootChain.approveToken({
    erc20Address: erc20Deposit.currency,
    amount: erc20Deposit.amount,
    txOptions: {
      from: erc20Deposit.address,
      privateKey: erc20Deposit.privateKey
    }
  });

  // deposit ERC20 funds
  const receipt = await rootChain.deposit({
    amount: erc20Deposit.amount,
    currency: erc20Deposit.currency,
    txOptions: {
      from: erc20Deposit.address,
      privateKey: erc20Deposit.privateKey,
      gas: gasLimit
    }
  });
}
```

- `gasLimit` - gas limit for your transaction. Please check the current data on Gas Station or similar resources.

Lifecycle

1. A user calls the `deposit` function that creates an RLP-encoded transaction that will be used to deposit into the ETH or ERC-20 `Vault` smart contracts (`ETHVault` for ETH and `ERC20Vault` for ERC20 tokens).
2. The `Vault` in question creates a deposit block and submits it to the `PlasmaFramework` contract.
3. The `Vault` in question emits a `DepositCreated` event.
4. The child chain receives the `DepositCreated` and creates the corresponding UTXO.
5. After a defined finality period the UTXO is ready for transacting on the network.

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 omg-js-samples repository:

```
git clone https://github.com/omgnetwork/omg-js-samples.git
```

2. Create `.env` file and provide the required configuration values.

3. Run these commands:

```
npm install
npm run start
```

4. Open your browser at http://localhost:3000.

5. Select `Make an ETH Deposit` or `Make an ERC20 Deposit` 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.