Documentation: Payment Validation Contract using OpenZeppelin Libraries

Introduction

The PaymentValidation contract is a sample Solidity code that uses OpenZeppelin libraries to validate the payment of a token and print the result of the transaction on the RSK Testnet blockchain.

Contract Functionality

The contract provides the following functionality:

- Constructor: Initializes the contract with the address of the payment token.

- `validatePayment` function: Validates the payment amount by checking the balance and allowance of the payment token for the sender. If the payment is valid, it emits a `PaymentVerified` event.

Deployment Instructions

To deploy the PaymentValidation contract on the RSK Testnet, follow the steps below:

Step 1: Install Dependencies

Make sure you have the necessary dependencies installed. In this example, we assume you are using Truffle framework to compile and deploy the contract. Run the following command in your project directory:

```shell
npm install @openzeppelin/contracts truffle truffle-hdwallet-provider@1.0.17
```

Step 2: Configure truffle-config.js

Create a `truffle-config.js` file in your project directory and configure it with the following content:

```javascript
const HDWalletProvider = require('truffle-hdwallet-provider');

const mnemonic = 'your twelve word mnemonic';
const infuraApiKey = 'your infura API key';

module.exports = {
networks: {
rskTestnet: {
provider: () => new HDWalletProvider(mnemonic, `https://public-node.testnet.rsk.co/$
{infuraApiKey}`),
network_id: 31,
gas: 6721975,
confirmations: 2,
timeoutBlocks: 200,
skipDryRun: true,
},
},
};
```

Make sure to replace `your twelve word mnemonic` with your actual mnemonic generated during wallet creation and `your infura API key` with your Infura API key.

Step 3: Create Migration File

Create a new migration file, for example, `2_deploy_contract.js`, in your project's migrations directory and add the following content:

```javascript
const PaymentValidation = artifacts.require('PaymentValidation');
const Token = 'your token address'; // Replace with the token address you want to use for payment validation

module.exports = function (deployer) {
deployer.deploy(PaymentValidation, Token);
};
```

Replace `your token address` with the actual address of the ERC20 token you want to use for payment validation.

Step 4: Run Migration

Execute the following command from your project directory to compile and deploy the contract on the RSK Testnet:

```shell
truffle migrate --network rskTestnet
```

This will deploy the contract on the RSK Testnet blockchain.

Interacting with the Contract

Once the contract is deployed, you can interact with it through a web interface or any other method you prefer. The contract provides the `validatePayment` function to validate the payment amount. Here's an example of how to use it:

```javascript
const paymentValidationContract = new web3.eth.Contract(abi, contractAddress); // Replace abi and contractAddress with actual values

// Call the validatePayment function with the required payment amount
```

```
const amount = 100; // replace with the desired payment amount
paymentValidationContract.methods.validatePayment(amount).send({ from: senderAddress }, function
(error, transactionHash) {
if (!error) {
console.log('Payment validation successful!');
console.log('Transaction hash:', transactionHash);
} else {
console.error('Payment validation error:', error);
}
});
```

Remember to replace `abi` and `contractAddress` with the actual values of the contract's ABI and address. Also, replace `senderAddress` with the address of the sender.

Conclusion

By following the deployment instructions and interacting with the PaymentValidation contract, you can validate payments using OpenZeppelin libraries on the RSK Testnet blockchain. Feel free to modify the contract or extend its functionality to suit your specific requirements.