



**Описание API-методов блокчейн-платформы
IZZIO**

ООО "Изио"
Москва, 2020

Введение	3
1. Node.js API	3
1.1. Класс NodeRPC	3
1.2. Класс EsmaSmartRPC	4
2. PHP API	6
2.1. Класс NodeRPC	6
2.2. Класс EsmaSmartRPC	8
3. JSON RPC API	10
3.1. Основные методы	10
4. API методы класса EsmaContract	11

Введение

API представляет собой классы, предназначенные для удаленного вызова основных процедур (от англ. **Remote Procedure Call, RPC**) блокчейна. Вызов происходит с помощью post и get запросов к серверу. Представлены несколько реализаций API: для использования в приложениях Node.js, для PHP приложений и для прямых вызовов JSON API.

1. Node.js API

Node.js биндинги API. Доступны в главном репозитории в `/api/node/`

1.1. Класс NodeRPC

Конструктор:

```
new NodeRPC([RPCUrl = 'http://localhost:3001/', pass = ''])
```

Аргументы:

- RPCUrl (string) - Адрес удаленного сервера, к которому происходит подключение в формате "<http://address:port>"
- pass (string) - Пароль для подключения к удаленному серверу

Returns object:

Проинициализированный объект с набором методов.

Http запрос к серверу:

```
_urlRequest([method = 'GET', url = "", params = [], password = "", login = "1337"])
```

Аргументы:

- method (string) - Метод выполнения запроса
- url (string) - Адрес к которому выполняется запрос
- params (array) - Массив параметров запроса (для POST)
- password (string) - Пароль(в случае, если подключение защищено паролем)
- login (string) - Логин для подключения

Returns Promise<object>:

Возвращает промис объекта ответа сервера на запрос

RPC запрос к серверу:

```
_request([method = "", params = [], paramStr = ""])
```

Аргументы:

- method (string) - Метод, вызываемый на сервере
- params (array) - Массив параметров запроса (для POST)

- `paramStr (string)` - Параметры, которые будут добавлены в URL запроса
- Returns Promise<object>:
Возвращает промис объекта с содержимым поля данных ответа от сервера

Получение информации и статуса блокчейна:

getInfo()

Returns Promise<object>:

Возвращает промис с информацией о блокчейне

Создание и регистрация нового кошелька:

createWallet()

Returns Promise<object>:

Возвращает промис с информацией о новом кошельке

Получение адреса текущего кошелька:

getWallet()

Returns Promise<string>:

Возвращает промис с адресом текущего кошелька

Изменение текущего кошелька на новый. Список транзакций при этом будет пересчитан, что может занять значительное время:

changeWalletByData(id, privateKey, publicKey)

Аргументы:

- `id (string)` - Адрес нового кошелька
- `privateKey (string)` - Приватный ключ для нового кошелька
- `publicKey (string)` - Публичный ключ для нового кошелька

Returns Promise<string>:

Возвращает промис со статусом выполнения операции

Получение блока по его id:

getBlockById(blockId)

Аргументы:

- `blockId (number | string)` - Идентификатор блока

Returns Promise<object>:

Возвращает промис с информацией о блоке

1.2. Класс `EcmaSmartRPC`

Данный класс предоставляет основной набор методов для работы с ECMA контрактами. Унаследован от `NodeRPC`.

Конструктор:

`new EcmaSmartRPC([RPCUrl = 'http://localhost:3001/', pass = ''])`

Аргументы:

- RPCUrl (string) - Адрес удаленного сервера, к которому происходит подключение в формате "<http://address:port>"
- pass (string) - Пароль для подключения к удаленному серверу

Returns object:

Проинициализированный объект с набором методов.

Получение информации о подсистеме смарт контрактов ЕСМА.

ecmaGetInfo()

Returns Promise<object>:

Возвращает промис объекта с информацией о подсистеме смарт контрактов ЕСМА.

Получение информации о контракте:

ecmaGetContractInfo(contractAddress)

Аргументы:

- contractAddress (string) - Адрес контракта, информацию о котором хотим получить

Returns Promise<object>:

Возвращает промис объекта с информацией о контракте.

Получение свойства контракта:

ecmaGetContractProperty(contractAddress, property)

Аргументы:

- contractAddress (string) - Адрес контракта, информацию о котором хотим получить
- property (string) - Имя свойства

Returns Promise<any>:

Возвращает промис с запрашиваемым свойством. Тип возвращаемого значения зависит от типа свойства.

Вызов метода контракта, с последующим откатом изменений:

ecmaCallMethod(contractAddress, method, params)

Аргументы:

- contractAddress (string) - Адрес контракта
- method (string) - Имя вызываемого метода
- params (array) - Параметры для метода

Returns Promise<any>:

Возвращает промис результата вызова или ошибку.

Вызов метода контракта, с последующей записью изменений в цепочку блоков:

ecmaDeployMethod(contractAddress, method, params)

Аргументы:

- contractAddress (string) - Адрес контракта
- method (string) - Имя вызываемого метода
- params (array) - Параметры для метода

Returns Promise<any>:

Возвращает промис с содержимым блока или ошибку.

Запуск нового контракта в сеть:

ecmaDeployContract(*source* [, *resourceRent* = '0'])

Аргументы:

- *source*(string) - Исходный JavaScript код смарт контракта
- *resourceRent* (string | number) - Количество токенов выделяемых для аренды ресурсов

Returns Promise<any>:

Возвращается объект контракта либо ошибка

Запуск нового контракта в сеть с подписанным блоком:

ecmaDeployContractSignedBlock(*block* [, *resourceRent* = '0'])

Аргументы:

- *block* (object) - JavaScript объект подписанного блока контракта
- *resourceRent* (string | number) - Количество токенов выделяемых для аренды ресурсов

Returns Promise<any>:

Возвращается объект контракта либо ошибка

Вызов метода контракта, с последующей записью изменений в цепочку блоков. В отличие от `deployMethod` принимает на вход подписанный блок `EcmaContractCallBlock`:

ecmaDeployMethodSignedBlock(*contractAddress*, *block*)

Аргументы:

- *contractAddress* (string) - Адрес контракта
- *block* (object) - JavaScript объект подписанного блока контракта

Returns Promise<any>:

Возвращает промис с содержимым блока или ошибку.

2. PHP API

API классы для PHP. Доступны в `/api/php`

2.1. Класс NodeRPC

Данный класс предоставляет базовый набор методов для реализации запросов к блокчейну.

Конструктор:

```
public function __construct([$RPCUrl = 'http://localhost:3001/', $password = ''])
```

Аргументы:

- *\$RPCUrl* (string) - Адрес удаленного сервера, к которому происходит подключение в формате "<http://address:port>"

- `$password` (string) - Пароль для подключения к удаленному серверу

Returns mixed:

Проинициализированный объект с набором методов.

cURL запрос к серверу:

private static function **curlRequest**([`$method` = 'get', `$url`, `$params` = [], `$password` = ''])

Аргументы:

- `$method` (string) - Метод выполнения запроса
- `$url` (string) - Адрес к которому выполняется запрос
- `$params` (array) - Массив параметров запроса (для POST)
- `$password` (string) - Пароль(в случае, если подключение защищено паролем)

Returns mixed | string:

Возвращает ответ сервера на запрос

RPC запрос к серверу:

protected function **request**([`$method`, `$params` = [], `$paramStr` = ''])

Аргументы:

- `$method` (string) - Метод, вызываемый на сервере
- `$params` (array) - Массив параметров запроса (для POST)
- `$paramStr` (string) - Параметры, которые будут добавлены в URL запроса

Returns array | mixed | `InvalidMethodException` | `ReturnException` | `RpcCallException` :

Возвращает тело ответа от сервера либо исключение

Получение информации и статуса блокчейна:

public function **getInfo**()

Returns mixed | `InvalidMethodException` | `ReturnException` | `RpcCallException` :

Возвращает объект с информацией о блокчейне либо исключение

Создание и регистрация нового кошелька:

public function **createWallet**()

Returns mixed | `InvalidMethodException` | `ReturnException` | `RpcCallException`:

Возвращает информацию о новом кошельке либо исключение

Получение адреса текущего кошелька:

public function **getWallet**()

Returns mixed | `InvalidMethodException` | `ReturnException` | `RpcCallException`:

Возвращает объект с адресом текущего кошелька либо исключение

Изменение текущего кошелька на новый. Список транзакций при этом будет пересчитан, что может занять значительное время:

public function **changeWalletByData**(`$id`, `$private`, `$public`)

Аргументы:

- `$id` (string) - Адрес нового кошелька
- `$privateKey` (string) - Приватный ключ для нового кошелька
- `$publicKey` (string) - Публичный ключ для нового кошелька

Returns array| `InvalidMethodException` | `ReturnException` | `RpcCallException`:

Возвращает новый кошелек либо исключение

Изменение текущего кошелька на новый. Список транзакций при этом будет пересчитан, что может занять значительное время:

```
public function changeWallet($wallet)
```

Аргументы:

- \$wallet (array) - Массив, полученный из метода createWallet

Returns array | InvalidMethodException | ReturnException | RpcCallException:

Возвращает новый кошелек либо исключение

2.2. Класс EcmaSmartRPC

Данный класс предоставляет основной набор методов для работы с ECMA контрактами. Унаследован от NodeRPC.

Конструктор:

```
public function __construct([$RPCUrl = 'http://localhost:3001/', $password = ''])
```

Аргументы:

- \$RPCUrl (string) - Адрес удаленного сервера, к которому происходит подключение в формате "<http://address:port>"
- \$password (string) - Пароль для подключения к удаленному серверу

Returns mixed:

Проинициализированный объект с набором методов.

Получение информации о подсистеме смарт контрактов ECMA.

```
public function ecmaGetInfo()
```

Returns array | mixed:

Возвращает объект с информацией о подсистеме смарт контрактов ECMA.

Получение информации о контракте:

```
public function ecmaGetContractInfo($contractAddress)
```

Аргументы:

- \$contractAddress (string) - Адрес контракта, информацию о котором хотим получить

Returns array | mixed:

Возвращает объекта с информацией о контракте.

Получение свойства контракта:

```
public function ecmaGetContractProperty($contractAddress, $property)
```

Аргументы:

- \$contractAddress (string) - Адрес контракта, информацию о котором хотим получить
- \$property (string) - Имя свойства

Returns array | mixed:

Возвращает объект с запрашиваемым свойством. Тип возвращаемого значения зависит от типа свойства.

Вызов метода контракта, с последующим откатом изменений:

```
public function ecmaCallMethod($contractAddress, $method, $params)
```

Аргументы:

- \$contractAddress (string) - Адрес контракта
- \$method (string) - Имя вызываемого метода
- \$params (array) - Параметры для метода

Returns array | mixed:

Возвращает результат вызова или ошибку.

Вызов метода контракта, с последующей записью изменений в цепочку блоков:

```
public function ecmaDeployMethod($contractAddress, $method, $params)
```

Аргументы:

- \$contractAddress (string) - Адрес контракта
- \$method (string) - Имя вызываемого метода
- \$params (array) - Параметры для метода

Returns array | mixed:

Возвращает содержимое блока или ошибку.

Запуск нового контракта в сеть:

```
public function ecmaDeployContract($source [, $resourceRent = '0'])
```

Аргументы:

- \$source(string) - Исходный JavaScript код смарт контракта
- \$resourceRent (string | number) - Количество токенов выделяемых для аренды ресурсов

Returns array | mixed:

Возвращается объект контракта либо ошибка

Запуск нового контракта в сеть с подписанным блоком:

```
public function ecmaDeployContractSignedBlock($block [, $resourceRent = '0'])
```

Аргументы:

- \$block (object) - JavaScript объект подписанного блока контракта
- \$resourceRent (string | number) - Количество токенов выделяемых для аренды ресурсов

Returns array | mixed:

Возвращается объект контракта либо ошибка

Вызов метода контракта, с последующей записью изменений в цепочку блоков. В отличие от deployMethod принимает на вход подписанный блок EcmaContractCallBlock:

```
public function ecmaDeployMethodSignedBlock($contractAddress, $block)
```

Аргументы:

- \$contractAddress (string) - Адрес контракта
- \$block (object) - JavaScript объект подписанного блока контракта

Returns array | mixed:

Возвращает содержимое блока или ошибку.

3. JSON RPC API

Встроенные RPC-API методы. Доступны на RPC порту узла. При не пустом значении rpcPassword требуется HTTP Auth с указанием пароля и любого произвольного пользователя

3.1. Основные методы

- **GET** /getInfo
- **GET** /getBlock/:id:
- **GET** /isReadyForTransaction
- **POST** /createWallet
- **POST** /resyncBlockchain
- **GET** /downloadWallet
- **POST** /restoreWallet
- **POST** /changeWallet

GET /getInfo

Получение информации о текущем узле блокчейн

GET /getBlock/:id:

Получение объекта блока по номеру

GET параметры:

- **id** - Строка. Index блока

GET /isReadyForTransaction

Возвращает true или false. Проверка завершенности синхронизации, готовность узла создавать новые транзакции.

POST /createWallet

Создает новый кошелек с генерацией приватного и публичного ключа. Возвращает структуру Wallet

POST /resyncBlockchain

Запуск пересчёта цепочки блоков

GET /downloadWallet

Возвращает файл текущего кошелька с публичным и приватным ключом

POST /restoreWallet

Загружает структуру кошелька как основную

POST параметры:

- **public** - Строка. Публичный ключ кошелька
- **private** - Строка. Приватный ключ кошелька
- **id** - Строка. Адрес кошелька
- **block** - Число. Номер блока в котором определен кошелек
- **balance** - Число. Баланс кошелька

POST /changeWallet

Аналогично restoreWallet

POST параметры:

- **public** - Строка. Публичный ключ кошелька
- **private** - Строка. Приватный ключ кошелька
- **id** - Строка. Адрес кошелька
- **block** - Число. Номер блока в котором определен кошелек
- **balance** - Число. Баланс кошелька

4. API методы класса EcmaContract

Методы contracts/ecma предоставляют функционал работы со смарт-контрактами EcmaContract

Для взаимодействия с контрактом с помощью API используются методы:

- **POST** contracts/ecma/deployContract
- **GET** contracts/ecma/getContractInfo/:contractAddress:
- **GET** contracts/ecma/getContractProperty/:contractAddress/:property:
- **POST** contracts/ecma/callMethod/:contractAddress/:method:
- **POST** contracts/ecma/deployMethod/:contractAddress/:method:
- **POST** contracts/ecma/deploySignedMethod/:contractAddress:
- **GET** /contracts/ecma/getInfo

POST contracts/ecma/deployContract

Запуск нового контракта в сеть

POST параметры:

- **resourceRent** - Число (от 0 и более). Количество токенов выделяемых для аренды ресурсов
- **source** - Строка. Исходный JavaScript код смарт контракта

POST contracts/ecma/deployContract *Альтернативный способ вызова*

Запуск нового контракта в сеть с помощью подписанного блока

POST параметры:

- **resourceRent** - Число (от 0 и более). Количество токенов выделяемых для аренды ресурсов
- **source** - JSON объект, содержащий подписанный блок запуска контракта

GET contracts/ecma/getContractInfo/:contractAddress

Получение информации о контракте

GET параметры:

- **contractAddress** - Строка. Адрес контракта

GET contracts/ecma/getContractProperty/:contractAddress/:property

Получение значения свойства контракта

GET параметры:

- **contractAddress** - Строка. Адрес контракта
- **property** - Строка. Название свойства

POST contracts/ecma/callMethod/:contractAddress/:method:

Вызов метода контракта, с последующим откатом изменений, и возвратом результата вызова или ошибки

GET параметры:

- **contractAddress** - Строка. Адрес контракта
- **method** - Строка. Название вызываемого метода

POST параметры:

- **argsEncoded** - Строка. JSON сериализованное значение со списком аргумента вызова

POST contracts/ecma/deployMethod/:contractAddress/:method:

Вызов метода контракта, с последующей записью изменений в цепочку блоков.

Возвращает содержимое нового блока или ошибку.

GET параметры:

- **contractAddress** - Строка. Адрес контракта
- **method** - Строка. Название вызываемого метода

POST параметры:

- **argsEncoded** - Строка. JSON сериализованное значение со списком аргумента вызова

POST contracts/ecma/deploySignedMethod/:contractAddress:

Вызов метода контракта, с последующей записью изменений в цепочку блоков. В отличие от deployMethod принимает на вход подписанный блок EcmaContractCallBlock

GET параметры:

- **contractAddress** - Строка. Адрес контракта

POST параметры:

- **source** - Строка. JSON сериализованный объект подписанного блока

GET /contracts/ecma/getInfo

Получение информации о последнем блоке сети

