Спецификация публичного программного интерфейса к тестовой блокчейн-платформе ПАО «Сбербанк»

Версия документа: 8d3da8 от 19.08.2021 г.

Оглавление

Введение

О платформе

Блокчейн-платформа Сбера реализована на базе open-source решения Hyperledger Fabric от The Linux Foundation, доработанного для задач токенизации активов и совершения сделок с ними. Функционал платформы позволяет участникам:

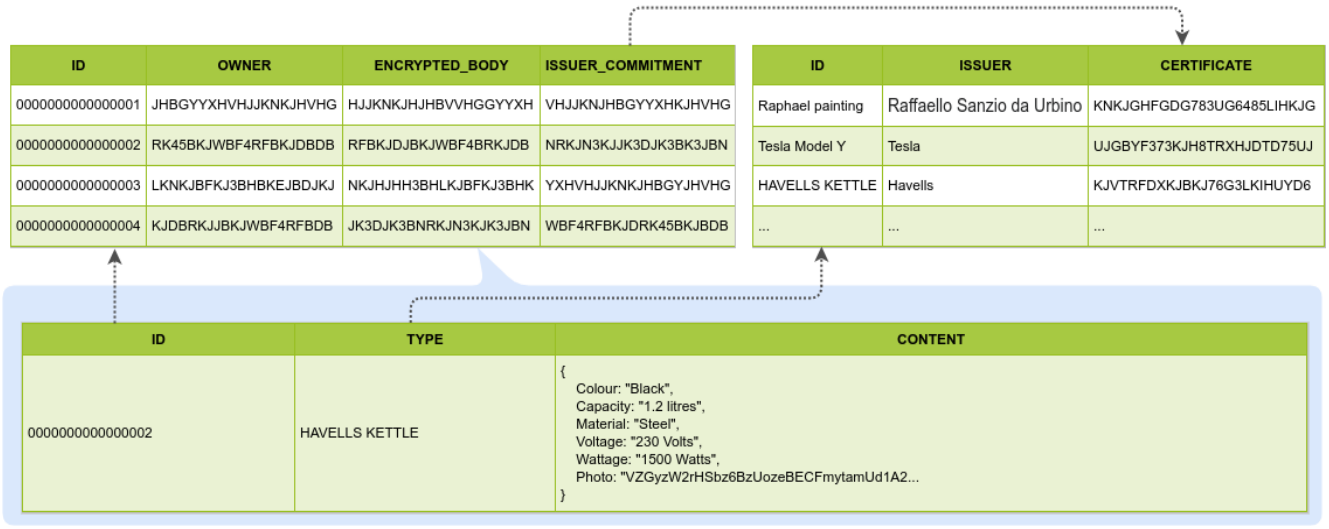
- регистрировать различные типы токенов;
- выпускать токены зарегистрированных типов;
- выставлять и принимать предложения на обмен токенов;
- атомарно обмениваться токенами;
- сжигать (погашать) токены;

Платформа интегрирована с автоматизированными системами ПАО «Сбербанк», что позволяет связать исполнение смарт-контрактов с платежами в рублях.

Понятие токена на платформе

CNFT – Confidential **N**on-**F**ungible **T**oken (конфиденциальный невзаимозаменяемый токен). Цифровая сущность, описывающая уникальный актив и факт владения им, соблюдая при это конфиденциальность или банковскую тайну. Токенезации может подлежать любой актив: имущество, товары, ценные бумаги, денежные средства и т.д. Токен представляет собой комбинацию:

- уникального идентификатора;
- типа токена;
- некоторого произвольного содержимого;
- подписи этих атрибутов со стороны эмитента токена;



Структура данных CNFT 3.0

Чтобы стать эмитентом токенов участник платформы должен:

1. Создать тип токена с атрибутами, значения которых будут содержаться в блоке **CONTENT** токенов данного типа.
2. Тип токена связывается с публичным ключом (сертификатом) эмитента.
3. Создать экземпляр токена, указав значения его атрибутов.
4. При создании экземпляра токена эмитент подписывает зашифрованное содержимое поля **CONTENT** токена (**ENCRYPTED_BODY**) сертификатом, привязанным к типу данного токена.
   Подпись эмитента содержится в поле **ISSUER_COMMITMENT**.

Структуры данных «Типы токенов» и «Владельцы токенов» хранятся в блокчейне. При этом владелец токена представлен одноразовым анонимным публичным ключом. Структура данных «Токен» хранится непосредственно его владельцами offchain. Данная схема позволяет:

- Вести децентрализованный распределенный реестр токенов и их владельцев, предотвращающий двойную трату токенов.

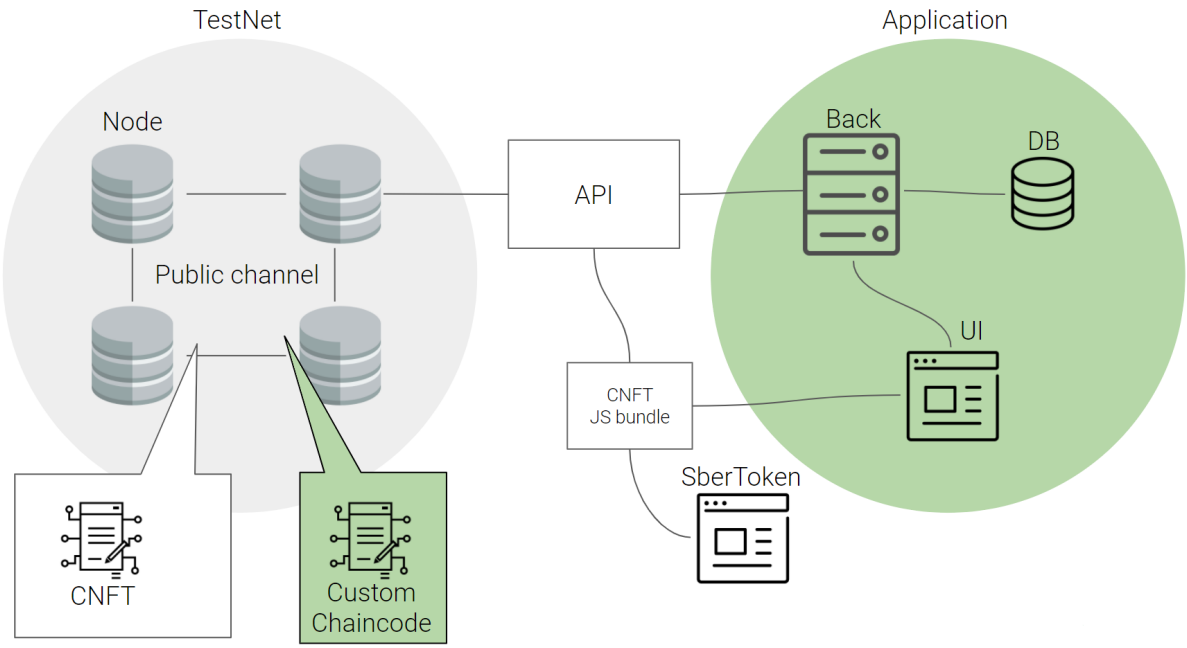
- Осуществлять валидацию транзакций независимыми участниками консенсуса блокчейн-сети.
- Осуществлять проверку аутентичности токена - факт его выпуска определенным эмитентом.
- Обеспечить конфиденциальность как владельца, так и содержимого токена.
- Проверить уникальность ID токена и, следовательно, уникальность самого токена;
- Чтобы содержание токена было конфиденциальным, участники должны договариваться о сделках, используя закрытые каналы коммуникации.

TestNet

Интеграция с платформой может осуществляться несколькими способами:

1. Используя JavaScript-библиотеку (описана в настоящем документе). Библиотека доступна в трех вариантах исполнения: **ECMAScript modules** (React), **CommonJS modules** (Node JS), **Standard** (экспорт в общий скоуп).
2. **JVM**-библиотека.
3. Работа напрямую с **REST-API** гейта.
4. Развёртывание собственного узла.

Для загрузки в тестовую сеть кастомизированного чейнкода необходимо обратиться в лабораторию блокчейн Сбера по адресу blockchain@sberbank.ru.



Спецификация программного интерфейса

Инициализация

Создание криптографической конфигурации

```
let cryptoConfig =
  new CryptographyConfiguration(
    CNFTCrypto.webCryptoSign(), // Issuer operations
    CNFTCrypto.webCryptoSign(), // Token operations
    CNFTCrypto.webCryptoSign(), // Identity operations
    CNFTCrypto.webCryptoEncryption() // Encryption operations
  );
```

Создание конфигурации хранилища

localStorage – хранилище внутри браузера.

inMemory – хранилище в памяти.

Пример создания конфигурации хранилища:

```
let store = CNFTStore.localStorage();
```

Создание списка методов для прослушивания событий и сообщений (walletListener)

onTokenTypeRegistered*(wallet, tokenTypes)*, где wallet – экземпляр кошелька-получателя события, tokenTypes массив структур TokenType.

onTokenListChanged*(wallet, tokensAdded, tokensIdsRemoved)*, где *wallet* – экземпляр кошелька-получателя события, *tokensAdded* и *tokensIdsRemoved* массив идентификаторов токенов (tokenId) в случае добавления токена в список он будет в массиве *tokensAdded*, в случае удаления из списка токенов в *tokensIdsRemoved.*

onIssuedTokensUpdated *(wallet, addedSeq, swappedSeq, burnedSeq)*, где *wallet* – экземпляр кошелька-эмитента токена, *addedSeq* – массив структур SignedToken, *swappedSeq* – массив структур DealRequest, *burnedSeq* – массив структур BurntIssuedTokens.

onChangeRequested*(wallet, requests)*, где *wallet* – экземпляр кошелька-владельца токена, *requests* – массив структур ChangeOperationRequest.

onOffersChanged*(wallet, newOffers, closedOffers)*, где *wallet* – экземпляр кошелька-получателя, *newOffers* – массив структур Offer, *closedOffers* - строковый массив идентификаторов сделок (предложений).

onTokenRequested*(wallet, requests)*, где wallet - экземпляр кошелька-получателя, requests – массив сообщений структур TokenRequest.

onTransferProposed*(wallet, requests)*, где wallet - экземпляр кошелька-получателя, requests – массив сообщений структур TransferProposal.

onIssueRequested*(wallet, requests)*, где wallet - экземпляр кошелька-получателя, requests – массив сообщений структур IssueRequest.

onApplyForOffer *(wallet, requests)*, где wallet - экземпляр кошелька-получателя, requests – массив сообщений структур ApplyForOffer.

onApproveOffer*(wallet, requests)*, где wallet - экземпляр кошелька-получателя, requests – массив сообщений структур ApproveOffer.

Создании экземпляра кошелька

GATE_URL – адрес REST API гейта.

Важно! Далее под **wallet** будет подразумеваться экземпляр кошелька.

Пример создания экземпляра кошелька:

```
let wallet = await CNFT.newWallet(
  {
    "gate": CNFT.newGateApi(GATE_URL),
    "store" : CNFTStore.localStorage(),
    "crypto": cryptoConfig,
    "listener": walletListener
  } );
```

Методы

Операции с типами токенов

registerTokenType

Регистрация нового типа токена со случайным уникальным идентификатором.

- **Параметры вызова:**

Наименование	Тип данных	Описание
tokenMeta	*array [TokenFieldMeta]*	Коллекция объектов типа TokenFieldMeta

- **Пример вызова:**

```
const tokenMeta = [new TokenFieldMeta("test", TokenFieldType.Text)];

let myNewTokenType = await wallet.registerTokenType(tokenMeta);
```

- **Возврат:**

  Строковый идентификатор типа токена.

- **Событие:**

  onTokenTypeRegistered

listTokenTypes

Возвращает список всех зарегистрированных типов токенов.

- **Параметры вызова:**

  Вызывается без параметров

- **Пример вызова:**

```
let tokenTypesInfo = await wallet.listTokenTypes();
```

- **Возврат:**

  listTokenTypes возвращает список объектов TokenTypeInfo**.**

Адреса и идентификаторы токенов

createAddress

Создает адрес, который потом может быть использован другим человеком для передачи токена.

- **Параметры вызова:**

  Вызывается без параметров

- **Пример вызова:**

```
let address = await wallet.createAddress();
```

- **Возврат:**

  createAddress возвращает адрес пользователя в виде **array buffer**.

reserveId

Резервирует идентификатор токена.

- **Параметры вызова:**

  Вызывается без параметров

- **Пример вызова:**

```
await wallet.reserveId();
```

- **Возврат:**

  reserveId возвращает объект ReservedId.

createIdAccess

Создаёт ключ для токена.

- **Параметры вызова:**

Наименование	Тип данных	Описание
tokenId	*string*	Уникальный идентификатор токена

- **Пример вызова:**

```
await wallet.createIdAccess(tokenId);
```

- **Возврат:**

  createIdAccess возвращает ключ токена в виде **array buffer**.

Операции с токенами

issue

Выпуск нового токена существующего типа со случайным уникальным идентификатором.

Важно! Выпуск токена может быть осуществлен, только эмитентом типа токена.

- **Параметры вызова:**

Наименование	Тип данных	Описание
tokenType	*string*	Уникальный идентификатор типа токена, указанный в ответе при выпуске токена, либо же при вызове метода ListTokenTypes (theType.typeId)
content	*array [string]*	Содержание токена зависит от типа токена и описывается TokenTypeInfo.tokenMeta.

- **Пример вызова:**

```
let tokenType = “abc”;

let tokenContent = [“text”];

let tokenBody = await wallet.issue(tokenType, tokenContent);
```

- **Возврат:**

  issue возвращает объект TokenBody (тело токена).

- **События:**

  onTokenListChanged, onIssuedTokensUpdated

issueFor

Выпуск нового токена существующего типа для другого пользователя.

- **Параметры вызова:**

Наименование	Тип данных	Описание
memberId	*string*	Идентификатор кошелька кому выпускается токен
tokenId	*string*	Публичный идентификатор, соответствующий данному токену (tokenId).
idAccess	*array buffer*	Ключ доступа к уникальному идентификатору токена, позволяющий осуществить выпуск.
tokenType	*string*	Уникальный идентификатор типа токена, указанный в TokenType.typeId.
content	*array [string]*	Содержание токена зависит от типа токена и описывается TokenType.tokenMeta.

- **Пример вызова:**

```
await wallet.issueFor(memberId, tokenId, idAccess, tokenType, content);
```

- **Возврат:**

  issueFor возвращает объект TokenBody (тело токена).

listTokens

Возвращает список всех токенов пользователя.

- **Параметры вызова:**

  Вызывается без параметров.

- **Пример вызова:**

```
await wallet.listTokens();
```

- **Возврат:**

  listTokens возвращает коллекцию объектов SignedToken (объект подписанного токена).

getToken

Возвращает токен по идентификатору.

- **Параметры вызова:**

Наименование	Тип данных	Описание
tokenId	*string*	Публичный идентификатор, соответствующий данному токену.

- **Пример вызова:**

```
await wallet.getToken(tokenId);
```

- **Возврат:**

  getToken опционально возвращает объект SignedToken - объект подписанного токена.

getIssuedToken

Возвращает выпущенный токен по его идентификатору.

- **Параметры вызова:**

Наименование	Тип данных	Описание
tokenId	*string*	Публичный идентификатор, соответствующий данному токену.

- **Пример вызова:**

```
await wallet.getIssuedToken(tokenId);
```

- **Возврат:**

  getIssuedToken опционально возвращает объект SignedToken - объект подписанного токена.

requestChange

Запрос на изменение токена к владельцу токена.

- **Параметры вызова:**

Наименование	Тип данных	Описание
tokenId	*string*	Публичный идентификатор, соответствующий токену.
content	*array [string]*	Содержание токена зависит от типа токена и описывается TokenType.tokenMeta.
extraData	*array buffer*	Произвольный набор данных.

- **Пример вызова:**

```
await wallet.requestChange(tokenId, content, extraData);
```

- **Возврат:**

  requestChange возвращает коллекцию объектов TokenChangeRequest.

Передача токенов

sendToken

Передача токена.

- **Параметры вызова:**

Наименование	Тип данных	Описание
memberId	*string*	Идентификатор кошелька кому передаются токены
dealId	*string*	Произвольный идентификатор сделки
tokenIds	*array [string]*	Коллекция публичный идентификатор, соответствующий отправляемому токену.
to	*array buffer*	Публичный ключ получателя токена.

- **Пример вызова:**

```
await wallet.sendToken(memberId, dealId, tokenIds, to);
```

- **Возврат:**

  Метод ничего не возвращает.

Публичные предложения

putOffer

Публикует публичное предложение (оффер), оффер содержит предложение обменять один токен на другой. Токены в оффере не указаны, указано описание. Описание может быть не полным (содержать только существенные условия).

Например:

Есть тип токена "велосипед" атрибуты: "марка", "цвет".
Есть тип токена "Рубли".

Можно выставить офер: меняю "велосипед" на "рубли", можно "велосипед"/"ямаха" на "рубли", можно "велосипед"/"ямаха" /"красный" на "рубли"/"9999".

- **Параметры вызова:**

Наименование	Тип данных	Описание
supply	*TokenDescription*	Описание токена - предожение. Что меняется. Объект типа TokenDescription.
demand	*TokenDescription*	Описание токена - цена. На что меняется. Объект типа TokenDescription.

- **Пример вызова:**

```
const supplyTokenContent = [new TokenFieldValue(0, “test”)];

const demandTokenContent = [new TokenFieldValue(0, “test1”)];

const supply = new TokenDescription(“abc”, supplyTokenContent);

const demand = new TokenDescription(“bca”, demandTokenContent);

await wallet.putOffer(supply, demand);
```

- **Возврат:**

  putOffer возвращает объект типа ***Offer***.

- **Событие:**

  onOffersChange

listTokenSupplyCandidates

Возвращает список токенов в кошельке подходящих под оффер в качестве предложения или ***supply*** из Offer. Так как оффер может не иметь строгого описания, к нему могут подходить несколько токенов.

- **Параметры вызова:**

Наименование	Тип данных	Описание
offerId	*string*	Идентификатор оффера.

- **Пример вызова:**

```
await wallet.listTokenSupplyCandidates(offerId);
```

- **Возврат:**

  listTokenSupplyCandidates возвращает список объектов типа SignedToken.

listTokenDemandCandidates

Возвращает список токенов в кошельке подходящих под оффер в качестве цены или ***demand*** из Offer. Так как оффер может не иметь строгого описания, к нему могут подходить несколько токенов.

- **Параметры вызова:**

Наименование	Тип данных	Описание
offerId	*string*	Идентификатор оффера.

- **Пример вызова:**

```
await wallet.listTokenDemandCandidates(offerId);
```

- **Возврат:**

  listTokenDemandCandidates возвращает список объектов типа ***SignedToken***.

applyForOffer

Отправляет отклик на оффер через "систему сообщений" (сообщение публикуется в блокчейне для определенного адресата в зашифрованном виде).

- **Параметры вызова:**

Наименование	Тип данных	Описание
offerId	*string*	Идентификатор оффера
signedToken	*SignedToken*	Объект типа SignedToken

- **Пример вызова:**

```
await wallet.applyForOffer(offerId, signedToken);
```

- **Возврат:**

  Метод ничего не возвращает.

- **Сообщение:**

  onApplyForOffer

approveOffer

Принять предложение на оффер. Получив уведомление об отклике на оффер, "продавец" может принять предложение вызовом данного метода. В этот метод он передаёт конкретный токен, т.к. в офере токен задан в виде описания. У "продавца" также может быть несколько кандидатов, получить их можно вызовом listOfferSupplyCandidates.

- **Параметры вызова:**

Наименование	Тип данных	Описание
offerId	*string*	Идентификатор оффера
buyerOrgName	*string*	Идентификатор «покупателя», приславшего отклик на оффер
signedToken	*SignedToken*	Объект типа SignedToken

- **Пример вызова:**

```
await wallet.approveOffer(offerId, buyerOrgName, signedToken);
```

- **Возврат:**

  Метод ничего не возвращает.

- **Сообщение:**

  onApproveOffer

finalizeOffer

Совершение сделки по офферу. После того как "продавец" утвердил предложение "покупателя" и показал какой именно токен он продаёт при помощи метода approveOffer, покупатель может совершить сделку.

- **Параметры вызова:**

Наименование	Тип данных	Описание
offerId	*string*	Идентификатор оффера

- **Пример вызова:**

```
await wallet.finalizeOffer(offerId);
```

- **Возврат:**

  Метод ничего не возвращает.

closeOffer

Убирает оффер из публикации в открытом доступе. Оффер может висеть открытым сколько угодно времени и по нему может быть совершено сколько угодно сделок, в любой момент продавец может решить убрать офер, для этого используется данный метод.

- **Параметры вызова:**

Наименование	Тип данных	Описание
offerId	*string*	Идентификатор оффера

- **Пример вызова:**

```
await wallet.closeOffer(offerId);
```

- **Возврат:**

  Метод ничего не возвращает.

- **Сообщение:**

  onOffersChanged

listOffers

Возвращает список всех опубликованных в открытом доступе офферов.

- **Параметры вызова:**

  Вызывается без параметров.

- **Пример вызова:**

```
await wallet.listOffers();
```

- **Возврат:**

  listOffers возвращает список объектов типа WalletOffer.

getOffer

Возвращает приватную информацию о оффере по его идентификатору.

- **Параметры вызова:**

Наименование	Тип данных	Описание
offerId	*string*	Идентификатор оффера.

- **Пример вызова:**

```
await wallet.getOffer(offerId);
```

- **Возврат:**

  getOffer возвращает объект типа WalletOffer.

Погашение токена

burnToken

Уничтожить токен.

- **Параметры вызова:**

Наименование	Тип данных	Описание
tokenId	*string*	Публичный идентификатор токена.
extraData	*array buffer*	Произвольный набор данных.

- **Пример вызова:**

```
await wallet.burnToken(tokenId, extraData);
```

- **Возврат:**

  Метод ничего не возвращает.

Обмен сообщениями

proposeToken

Отправляет участнику сообщение с предложением перевести ему токен.

- **Параметры вызова:**

Наименование	Тип данных	Описание
to	*string*	Получатель сообщения. Идентификатор пользователя, зарегистрированный в "Адресной книге".
tokenType	*string*	Уникальный идентификатор типа токена, указанный в TokenType.typeId.
tokenContent	*array [string]*	Содержание токена зависит от типа токена и описывается TokenType.tokenMeta.
extraData	*array buffer*	Произвольный набор данных.

- **Пример вызова:**

```
await wallet.proposeToken(to, tokenType, tokenContent, extraData);
```

- **Возврат:**

  Метод ничего не возвращает.

- **Сообщение:**

  onTransferProposed

requestToken

Отправляет участнику сообщение с просьбой, перевести вам токен. Это сообщение следует использовать для:

- Попросить кого-либо прислать вам токен.
- Ответить на proposeToken.
- Запрос эмитента на выпуск токена.

- **Параметры вызова:**

Наименование	Тип данных	Описание
from	*string*	Получатель сообщения. Идентификатор пользователя, зарегистрированный в "Адресной книге".
tokenType	*string*	Уникальный идентификатор типа токена, указанный в TokenType.typeId.
tokenContent	*array [string]*	Содержание токена зависит от типа токена и описывается TokenType.tokenMeta.
extraData	*array buffer*	Произвольный набор данных.

- **Пример вызова:**

```
await wallet.requestToken(from, tokenType, tokenContent, extraData);
```

- **Возврат:**

  Метод ничего не возвращает.

- **Сообщение**

  onTokenRequested

requestIssue

Отправляет эмитенту сообщение с просьбой выпустить вам токен.

- **Параметры вызова:**

Наименование	Тип данных	Описание
tokenType	*string*	Уникальный идентификатор типа токена, указанный в TokenType.typeId.
tokenContent	*array [string]*	Содержание токена зависит от типа токена и описывается TokenType.tokenMeta.
extraData	*array buffer*	Произвольный набор данных.

- **Пример вызова:**

```
await wallet.requestIssue(tokenType, tokenContent, extraData);
```

- **Возврат:**

  Метод ничего не возвращает.

- **Сообщение:**

  onIssueRequested

Адресная книга

getIdentity

Возвращает текущего авторизованного пользователя.

- **Параметры вызова:**

  Вызывается без параметров.

- **Пример вызова:**

```
await wallet.getIdentity;
```

- **Возврат:**

  getIdentity возвращает идентификатор кошелька со стороны, которого был выполнен запрос.

listMembers

Получить список всех зарегистрированных участников.

- **Параметры вызова:**

  Вызывается без параметров.

- **Пример вызова:**

```
await wallet.listMembers;
```

- **Возврат:**

  listMembers возвращает список объектов типа MemberInformation**.**

CNFTHelper

jsOption

Обрабатывает опциональные ответы

- **Параметры вызова:**

Наименование	Тип данных	Описание
response	*option [Any]*	Опциональный ответ

- **Пример вызова:**

```
CNFTHelper.jsOption(optionResponse);
```

- **Возврат:**

JsOption в случае наличия данных возвращает их, в случае отсутствия ***undefined***.

Приложение: структуры данных

TokenType

Тип токена.

Наименование	Тип	Описание
typeId	*string*	Уникальный идентификатор типа токена.
tokenMeta	*array[TokenFieldMeta]*	Коллекция объектов типа TokenFieldMeta
issuerPublicKey	*array buffer*	Публичный ключ пользователя, зарегистрировавшего данный тип токена.
issuerId	*string*	Уникальный идентификатор пользователя, зарегистрировавшего данный тип токена.

TokenFieldMeta

Мета-информация типа токена.

Наименование	Тип	Описание
name	*string*	Наименование мета-информации типа токена.
typeId	*string*	Строковый идентификатор типа данных (рекомендуется использовать через указание константы в TokenFieldType)

TokenOwner

Владелец токена.

Наименование	Тип	Описание
require	*number*	Количество требуемых подписей (от ключей из `*keys*`) для совершения операции с токеном.
keys	*array buffer*	Коллекция публичных ключей владельцев токена в кодировке `*Base64*`

ReservedId

Зарезервированный идентификатор токена.

Наименование	Тип	Описание
tokenId	*string*	Публичный идентификатор, соответствующий данному токену.
owner	*TokenOwner*	Объект типа TokenOwner

TokenBody

Тело токена.

Наименование	Тип	Описание
tokenId	*string*	Публичный идентификатор, соответствующий данному токену.
tokenType	*string*	Уникальный идентификатор типа токена, указанный в TokenType.typeId.
content	*array [string]*	Содержание токена зависит от типа токена и описывается TokenType.tokenMeta.

SignedToken

Подписанный токен.

Наименование	Тип	Описание
tokenBody	*TokenBody*	Объект типа TokenBody
signature	*array buffer*	Подпись для байтового представления токена, созданная эмитентом.

IssuedToken

Выпущенный токен.

Наименование	Тип	Описание
signedToken	*array buffer*	Зашифрованное содержимое, подписанного токена
issuerCommitment	*array buffer*	Подпись эмитента
regulation	*string*	Информация о контроле и полномочиях регуляторов (для токена)

Token

Описание токена.

Наименование	Тип	Описание
id	*ReservedId*	Объект типа ReservedId.
token	*IssuedToken*	Объект типа IssuedToken

MemberInformation

Информация об участнике.

Наименование	Тип	Описание
name	*string*	Логин или имя участника.
signingPublic	*array buffer*	Публичный ключ для идентификации участника.
encryptionPublic	*array buffer*	Публичный ключ участника для шифрования.

Offer

Публичное предложение.

Наименование	Тип	Описание
id	*string*	Идентификатор оффера.
owner	*string*	Владелец оффера.
supply	*TokenDescription*	Описание токена - предложение. Что меняется. Объект типа TokenDescription.
demand	*TokenDescription*	Описание токена - цена. На что меняется. Объект типа TokenDescription.

TokenDescription

Описание параметров токена для формирования предложения.

Наименование	Тип	Описание
tokenType	*string*	Тип токена.
content	*array[TokenFieldValue]*	Коллекция TokenFieldValue

TokenFieldValue

Значение атрибутов токена.

Наименование	Тип	Описание
index	*number*	Индекс параметра.
value	*string*	Значение параметра.

DealRequest

Запрос на атомарный обмен.

Наименование	Тип	Описание
deal	*Deal*	Сделка. Объект типа Deal.
signatures	*array[DealSignatures]*	Коллекция подписей для заключения сделки (DealSignatures).

Deal

Сделка.

Наименование	Тип	Описание
dealId	*string*	Идентификатор сделки.
changes	*map<string, TokenChange>*	Идентификатор текущего владельца -> новый владелец, объект типа TokenChange.

TokenChange

Информация о новом владельце токена.

Наименование	Тип	Описание
tokenOwner	*TokenOwner*	Объект типа TokenOwner.
signedToken	*array buffer*	Зашифрованный SignedToken.

BurntIssuedTokens

Погашение токенов.

Наименование	Тип	Описание
signedToken	*SignedToken*	Объект типа SignedToken.
data	*array buffer*	Данные для погашения токена.

ChangeOperationRequest

Запрос размена токена

Наименование	Тип	Описание
signedBurnRequest	*SignedBurnRequest*	Объект типа SignedBurnRequest.
signedToken	*SignedToken*	Объект типа SignedToken.
extra	*BurnExtraData*	Объект типа BurnExtraData

TokenRequest

Запрос токена.

Наименование	Тип	Описание
tokenType	*string*	Тип токена.
content	*array[string]*	Контент для токена.
address	*TokenOwner*	Объект типа TokenOwner
extraData	*array buffer*	Произвольный набор данных.

TransferProposal

Запрос на перевод токена.

Наименование	Тип	Описание
tokenType	*string*	Тип токена.
content	*array[string]*	Контент для токена.
extraData	*array buffer*	Произвольный набор данных.

IssueRequest

Запрос на выпуск токена.

Наименование	Тип	Описание
token	*TokenBody*	Объект типа TokenBody
idAccess	*array buffer*	Ключ доступа к уникальному идентификатору токена.
extraData	*array buffer*	Произвольный набор данных.

ApplyForOffer

Подтверждение предложения со стороны покупателя.

Наименование	Тип	Описание
offerId	*string*	Идентификатор оффера.
signedToken	*SignedToken*	Объект типа *SignedToken*.
buyerTokenPublicKey	*array buffer*	Публичный ключ покупателя.
buyerOrgName	*string*	Индентификатор покупателя.

ApproveOffer

Подтверждение предложения со стороны продавца.

Наименование	Тип	Описание
offerId	*string*	Идентификатор оффера.
signedToken	*SignedToken*	Объект типа *SignedToken*.
dealRequest	*DealRequest*	Объект типа *DealRequest*.

TokenTypeInfo

Информация о типе токена.

Наименование	Тип	Описание
theType	*TokenType*	Объект типа *TokenType*.
owned	*bool*	Признак принадлежности тому, кто делал запрос

TokenChangeRequest

Запрос на измение/размен токена.

Наименование	Тип	Описание
tokenBody	*TokenBody*	Объект типа *TokenBody*.
idAccess	*array buffer*	Ключ доступа к уникальному идентификатору токена.

WalletOffer

Публичное предложение.

Наименование	Тип	Описание
offer	*Offer*	Объект типа *Offer*.
offerCandidate	*array[OfferCandidate]*	Коллекция типа *OfferCandidate*

SignedBurnRequest

Подписанный запрос на погашение токена.

Наименование	Тип	Описание
request	*BurnRequest*	Объект типа *BurnRequest*.
signatures	*array[BurnRequestSignatures]*	Коллекция типа *BurnRequestSignatures*

BurnExtraData

Данные для погашения токена.

Наименование	Тип	Описание
ownerIds	*array [string]*	Коллекция строковых идентификатор владельцев.
changeRequests	*array[TokenChangeRequest]*	Коллекция типа *TokenChangeRequest*
data	*array buffer*	Произвольный набор данных.

OfferCandidate

Одобренная сделка со стороны продавца.

Наименование	Тип	Описание
signedTokenBuyer	*SignedToken*	Объект типа SignedToken.
buyerEncryptionKey	*array buffer*	Идентификатор ключа шифрования покупателя.
buyerTokenKey	*array buffer*	Публичный ключ покупателя.
sellerApprove	*SellerApprove*	Объект типа *SellerApprove*.

DealSignatures

Список подписей к сделке.

Наименование	Тип	Описание
tokenId	*string*	Идентификатор токена.
dealSignature	*array buffer*	Подписи к сделке.
signForIssuers	*array buffer*	Зашифрованные подписи на ключе шифрования эмитента типа токена.

BurnRequest

Запрос на погашение токена.

Наименование	Тип	Описание
tokenId	*string*	Идентификатор токена.
extra	*array buffer*	Произвольный набор данных.

BurnRequestSignatures

Подписи для погашения.

Наименование	Тип	Описание
signatureToBurn	*array buffer*	Подписи для погашения токена.
burnSignatureForIssuer	*array buffer*	Зашифрованная подпись (на запрос погашения) на ключе эмитента.

SellerApprove

Подтверждения продавца на сделку.

Наименование	Тип	Описание
signedTokenSeller	*SignedToken*	Объект типа SignedToken
dealRequest	*DealRequest*	Объект типа DealRequest

Возвращаемые примитивные типы

- Array;
- Array Buffer;
- String;
- Bool;
- Number.