Документация Raydium Trade API

Введение

Raydium Trade API предоставляет возможность выполнения свопов токенов на блокчейне Solana. Этот документ содержит детальное описание каждого метода, функции и структуры ответа, используемых в API.

Основная функция: apiSwap

Функция `apiSwap` - это основной метод для обмена токенов.

Подключение библиотек

```
import web3 from '@solana/web3.js';
import { API_URLS, TokenAccount } from '@raydium-io/raydium-sdk-v2';
import { NATIVE_MINT } from '@solana/spl-token';
import { getPriorityFees, fetchTokenAccountData, getComputeSwapBaseIn,
postTransactionSwapBaseIn, signAndSendTransactions } from './api';
import { connection, getTxVersion, owner } from './constants';
import { PriorityFeeType, SwapComputeType, SwapTransactionsType } from './types';
```

Описание функции

Функция начинается с объявления переменных, которые используются для выполнения свопа.

- **inputMint** - адрес токена, который вы хотите обменять (входной токен).
- **outputMint** - адрес токена, который вы хотите получить (выходной токен).
- **amount** - количество входного токена для обмена в лампортах (наименьших единицах токена).
  Либо inputAmount, либо outpoutAmount в зависимости от режима обмена. Вам необходимо учитывать десятичные знаки (x10^n) например. Чтобы обменять 1 токенА, где токенА имеет 6 десятичных знаков, вам необходимо ввести 1 * 10^6 или 1 000 000.
- **slippage** - допустимое отклонение цены обмена в процентах. Допустимое отклонение в базовых пунктах (0,01%).
- **txVersion** - версия транзакции.

Логика работы

1. Определение версии транзакции.
2. Получение информации о токенах на счете с помощью `fetchTokenAccountData()`.
3. Проверка наличия аккаунта входного токена.
4. Получение приоритетной комиссии с помощью `getPriorityFees()`.
5. Получение данных о свопе через `getComputeSwapBaseIn()`.
6. Создание транзакции с помощью `postTransactionSwapBaseIn()`.
7. Подписание и отправка транзакций через `signAndSendTransactions()`.

Функция целиком

```
import web3 from '@solana/web3.js'
import { API_URLS, TokenAccount } from '@raydium-io/raydium-sdk-v2'
import { NATIVE_MINT } from '@solana/spl-token'
import {
  getPriorityFees,
  fetchTokenAccountData,
  getComputeSwapBaseIn,
  postTransactionSwapBaseIn,
  signAndSendTransactions,
} from './api'
import { connection, getTxVersion, owner } from './constants'
import { PriorityFeeType, SwapComputeType, SwapTransactionsType } from './types'

export const apiSwap = async () => {
  try {
    // это адрес токена, который вы хотите обменять (входной токен).
    const inputMint = NATIVE_MINT.toBase58()
    // это адрес токена, который вы хотите получить (выходной токен).
    const outputMint = '4k3Dyjzvzp8eMZWUXbBCjEvwSkkk59S5iCNLY3QrkX6R' // RAY
    // Этот параметр указывает количество входного токена для обмена.
    /*
      Это значение указывается в "лампортах" (наименьших единицах токена).
      Для SOL 1 лампорт =  10^-9 SOL
      Например, если хотите обменять 0.01 SOL,
      то значение должно быть 10000000 (10 миллионов лампортов).
    */
    const amount = 10000
    // Слиппейдж (slippage) — допустимое отклонение цены обмена.
    /*
      Это значение указывает на процент допустимого отклонения.
      Например, 0.5% означает, что цена свопа может отклониться на 0.5% от
расчетной.
```

```
*/
const slippage = 0.5
// Этот параметр указывает на версию транзакции.
const txVersion = getTxVersion()
const isV0Tx = txVersion === 'V0'
const isInputSol = inputMint === NATIVE_MINT.toBase58()
const isOutputSol = outputMint === NATIVE_MINT.toBase58()
const { tokenAccounts } = await fetchTokenAccountData()

const inputTokenAcc = tokenAccounts.find(
  (tokenAccount: TokenAccount) =>
    tokenAccount.mint.toBase58() === inputMint,
)?.publicKey

const outputTokenAcc = tokenAccounts.find(
  (tokenAccount: TokenAccount) =>
    tokenAccount.mint.toBase58() === outputMint,
)?.publicKey

if (!inputTokenAcc && !isInputSol) {
  console.error('ERROR: do not have input token account')
  return
}

const priorityFees: PriorityFeeType = await getPriorityFees(
  `${API_URLS.BASE_HOST}${API_URLS.PRIORITY_FEE}`,
)

console.log('priorityFees', JSON.stringify(priorityFees, null, 2))

const swapCompute: SwapComputeType = await getComputeSwapBaseIn({
  inputMint,
  amount,
  outputMint,
  slippage,
  txVersion,
})

console.log('swapCompute', JSON.stringify(swapCompute, null, 2))

const computeUnitPriceMicroLamports = String(priorityFees.data.default.h)

const swapTransactions: SwapTransactionsType =
  await postTransactionSwapBaseIn({
    computeUnitPriceMicroLamports,
    isInputSol,
```

```
        isOutputSol, // true means output mint receive sol, false means output mint
received wsol
        ownerPublicKey: owner.publicKey.toBase58(),
        swapCompute,
        txVersion,
        inputAccount: isInputSol ? undefined : inputTokenAcc?.toBase58(),
        outputAccount: isOutputSol ? undefined : outputTokenAcc?.toBase58(),
      })

    console.log('swapTransactions', JSON.stringify(swapTransactions, null, 2))

    const allTxBuf = swapTransactions.data.map((tx) =>
      Buffer.from(tx.transaction, 'base64'),
    )

    const allTransactions = allTxBuf.map((txBuf) =>
      isV0Tx
        ? web3.VersionedTransaction.deserialize(txBuf)
        : web3.Transaction.from(txBuf),
    )

    console.log(
      `total ${allTransactions.length} transactions`,
      swapTransactions,
    )

    await signAndSendTransactions({
      allTransactions,
      isV0Tx,
    })
  } catch (error) {
    console.error('ERROR in apiSwap:', error)
    throw error
  }
}
```

Ответы и структуры данных

Ответ getPriorityFees

Функция возвращает JSON-объект следующей структуры:

```
{
  "id": "0456352f-5102-47a5-8fd9-5a2bc37de0a3",
  "success": true,
  "data": {
    "default": {
      "vh": 20146,
      "h": 15109,
      "m": 10000
    }
  }
}
```


- **id** - уникальный идентификатор запроса.
- **success** - статус запроса (true/false).
- **data** - объект с приоритетной комиссией:
  - **vh** - комиссия при высокой приоритетности.
  - **h** - комиссия при средней приоритетности.
  - **m** - минимальная комиссия.

Ответ getComputeSwapBaseIn


```
{
  "id": "57765486-6669-49e7-b17c-3b75bccaa599",
  "success": true,
  "version": "V1",
  "data": {
    "swapType": "BaseIn",
    "inputMint": "So11111111111111111111111111111111111111112",
    "inputAmount": "10000",
    "outputMint": "4k3Dyjzvzp8eMZWUXbBCjEvwSkkk59S5iCNLY3QrkX6R",
    "outputAmount": "741",
    "slippageBps": 50,
    "priceImpactPct": 4.1,
    "referrerAmount": "0",
    "routePlan": [
      {
        "poolId": "E5TFaTumjuv1R9uHXzXWXztRxbu1jrkZJgszcd9KrnKr",
        "inputMint": "So11111111111111111111111111111111111111112",
        "outputMint": "HeLp6NuQkmYB4pYWo2zYs22mESHXPQYzXbB8n4V98jwC",
        "feeMint": "So11111111111111111111111111111111111111112",
        "feeRate": 100,
        "feeAmount": "100",
        "remainingAccounts": []
```

```
      },
      {
        "poolId": "ERbJqx2P5i2U9rP9eKtPNMyEhHTRjF5n1rkvRyrhQDWi",
        "inputMint": "HeLp6NuQkmYB4pYWo2zYs22mESHXPQYzXbB8n4V98jwC",
        "outputMint": "4k3Dyjzvzp8eMZWUXbBCjEvwSkkk59S5iCNLY3QrkX6R",
        "feeMint": "HeLp6NuQkmYB4pYWo2zYs22mESHXPQYzXbB8n4V98jwC",
        "feeRate": 18,
        "feeAmount": "12236",
        "remainingAccounts": ["H758QKqEGp3zURtXnMGzbe7bvtux3aL6rFBYM5ZuxrsH",
"CioZPte2bNWKfXg7FmWUFRKQJserMpvZeBeWzotPGM2y"],
        "lastPoolPriceX64": "176466728070810914691 8"
      }
    ]
  }
}
```


- **swapType** - тип обмена (BaseIn означает фиксированное количество входного токена).
- **inputMint** - адрес входного токена.
- **inputAmount** - количество входного токена.
- **outputMint** - адрес выходного токена.
- **outputAmount** - количество выходного токена после обмена.
- **slippageBps** - слиппейдж в базисных пунктах (0.5% = 50 BPS).
- **priceImpactPct** - влияние на цену в процентах.
- **referrerAmount** - количество токенов, которое получает реферер (обычно 0).
- **routePlan** - массив маршрутов обмена:
  - **poolId** - ID пула ликвидности.
  - **inputMint** - адрес входного токена пула.
  - **outputMint** - адрес выходного токена пула.
  - **feeMint** - токен, в котором взымается комиссия.
  - **feeRate** - ставка комиссии.
  - **feeAmount** - сумма комиссии.
  - **remainingAccounts** - дополнительные счета пула.
  - **lastPoolPriceX64** - последняя цена пула в формате X64.

Ответ postTransactionSwapBaseIn


```
{
  "id": "57765486-6669-49e7-b17c-3b75bccaa599-tx",
  "version": "V1",
  "success": true,
  "data": [
```

```
    {
      "transaction":
"AQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA..."
    }
  ]
}
```


- **id** - уникальный идентификатор транзакции.
- **version** - версия транзакции (например, V1).
- **success** - статус транзакции (true/false).
- **data** - массив транзакций с ключом:
  - **transaction** - базовая64 строка, содержащая данные транзакции.

Логи работы

Пример логов

1. Подключение к RPC-серверу:

```
[Raydium SDK] Connecting to RPC: https://api.mainnet-beta.solana.com (cluster:
mainnet)
```

2. Результат приоритетной комиссии:

```
priorityFees {
  "id": "0456352f-5102-47a5-8fd9-5a2bc37de0a3",
  "success": true,
  "data": { "vh": 20146, "h": 15109, "m": 10000 }
}
```

3. Результат свопа:

```
swapCompute { ... }
```

4. Завершение транзакции:

```
1 transaction sending..., txId:
2adsLh48q5zDzWRLzbEZaDSj9qqWffgWS1NvGMSUw1CJEDtk4NWB8GfUvGsDS48XsBSaNByidneFkgUgrR7
U3Khs
```