Документация Raydium SDK V2

Введение

Аллах свидетель.

Метод: `Raydium.load`

Метод `Raydium.load` инициализирует экземпляр Raydium SDK с указанными параметрами и опциями. Этот метод асинхронный и возвращает объект Raydium, который используется для дальнейшего взаимодействия с Raydium.

Сигнатура метода:

```
const raydium = await Raydium.load(config);
```

Параметры:

Метод принимает объект с параметрами, содержащий следующие ключи:

Параметр	Тип	Обязательный	Описание
`connection`	`Connection` (объект из web3.js)	Да	Соединение с блокчейном Solana через RPC URL.
`cluster`	`"mainnet" OR "devnet"`	Да	Сеть, к которой подключается SDK: `'mainnet'` или `'devnet'`.
`disableFeatureCheck`	`boolean`	Нет	Отключает проверку наличия определённых функций (по умолчанию `true`).
`blockhashCommitment`	`'processed' OR 'confirmed' OR 'finalized' OR 'recent' OR 'single' OR`	Нет	Коммитмент для транзакций: `'finalized'`, `'confirmed'`, `'processed'`.

Параметр	Тип	Обязательный	Описание
	`'singleGossip' OR 'root' OR 'max'`		Определяет уровень подтверждения транзакции.
`owner`	классы `Keypair` или `PublicKey`	Нет	Адрес кошелька пользователя или объект пары ключей (если процесс запускается на ноде).
`signAllTransactions`	`function`	Нет	Функция подписания всех транзакций. Может быть предоставлена библиотекой `@solana/wallet-adapter-react`.
`tokenAccounts`	`Array`	Нет	Массив токен-аккаунтов. Если децентрализованное приложение (dApp) обрабатывает это самостоятельно, можно передать готовые данные в SDK.
`tokenAccountRowInfos`	`Array`	Нет	Дополнительная информация о токен-аккаунтах. Если dApp обрабатывает это самостоятельно, можно передать напрямую.
`disableLoadToken`	`boolean`	Нет	По умолчанию `false`. Если не требуется загружать информацию о токенах, установить в `true`.

`config` имеет тип:

```
interface RaydiumLoadParams extends TokenAccountDataProp,
Omit<RaydiumApiBatchRequestParams, "api"> {
    connection: Connection;
    cluster?: Cluster;
    owner?: PublicKey | Keypair;
    apiRequestInterval?: number;
```

```
    apiRequestTimeout?: number;
    apiCacheTime?: number;
    signAllTransactions?: SignAllTransactions;
    urlConfigs?: API_URL_CONFIG;
    logRequests?: boolean;
    logCount?: number;
    jupTokenType?: JupTokenType;
    disableFeatureCheck?: boolean;
    disableLoadToken?: boolean;
    blockhashCommitment?: Commitment;
    loopMultiTxStatus?: boolean;
}
```

URL Конфигурации в Raydium SDK

Метод `Raydium.load` поддерживает передачу конфигураций API через объект `urlConfigs`. Это позволяет указать основные API-хосты и конечные точки для работы с различными компонентами протокола Raydium.

Параметры URL-конфигурации:

Параметр	Тип	Описание
`BASE_HOST`	`string`	Основной URL API. На текущий момент API не поддерживает `devnet`.
`OWNER_BASE_HOST`	`string`	URL для работы с владельцем активов.
`SERVICE_BASE_HOST`	`string`	URL для работы с сервисами Raydium.
`MONITOR_BASE_HOST`	`string`	URL для мониторинга.
`SERVICE_1_BASE_HOST`	`string`	URL для дополнительного сервиса.
`SEND_TRANSACTION`	`string`	URL для отправки транзакций.
`FARM_ARP`	`string`	URL для получения информации о доходности фермы (APR).
`FARM_ARP_LINE`	`string`	URL для получения графика APR по времени.
`CLMM_CONFIG`	`string`	URL для получения конфигураций концентрированного пула ликвидности.
`CPMM_CONFIG`	`string`	URL для получения конфигураций стандартного пула ликвидности.
`VERSION`	`string`	Версия API.
`CHECK_AVAILABILITY`	`string`	URL для проверки доступности сервера.
`RPCS`	`string`	URL для получения списка доступных RPC.

Параметр	Тип	Описание
`INFO`	`string`	URL для получения общей информации.
`STAKE_POOLS`	`string`	URL для получения списка стейкинг-пулов.
`CHAIN_TIME`	`string`	URL для получения текущего времени блокчейна.
`TOKEN_LIST`	`string`	URL для получения списка токенов.
`MINT_INFO_ID`	`string`	URL для получения информации о минте токена.
`JUP_TOKEN_LIST`	`string`	URL для получения списка токенов для Jupyter.

Конфигурация пулов:

Параметр	Тип	Описание
`POOL_LIST`	`string`	URL для получения списка пулов. Поддерживает параметры: - `poolType` : тип пула (`all` , `concentrated` , `standard` , `allFarm` , `concentratedFarm` , `standardFarm`) - `poolSortField` : поле сортировки (например, `liquidity` , `volume_24h` , `apr_24h`) - `sortType` : тип сортировки (`desc` , `asc`) - `page` : номер страницы - `pageSize` : количество записей на странице.
`POOL_SEARCH_BY_ID`	`string`	URL для поиска пула по ID: `?ids=idList.join(',')` .
`POOL_SEARCH_MINT`	`string`	URL для поиска пула по минту токена: - `mint1/mint2` : для поиска по парам токенов - `poolSortField` : критерий сортировки - `poolType` : тип пула - `sortType` : порядок сортировки - `page` , `pageSize` : пагинация.
`POOL_SEARCH_LP`	`string`	URL для поиска пула по LP токену: `?lps=lpList.join(',')` .
`POOL_KEY_BY_ID`	`string`	URL для получения ключа пула по ID: `?ids=idList.join(',')` .
`POOL_LIQUIDITY_LINE`	`string`	URL для получения линии ликвидности по ID пула: `?id=string` .
`POOL_POSITION_LINE`	`string`	URL для получения линии позиций пула.

Конфигурация ферм:

Параметр	Тип	Описание
`FARM_INFO`	`string`	URL для получения информации о ферме.
`FARM_LP_INFO`	`string`	URL для получения информации о фарминге по LP токену: `?lp=string&pageSize=100&page=number`.
`FARM_KEYS`	`string`	URL для получения ключей ферм.
`OWNER_CREATED_FARM`	`string`	URL для получения созданных ферм владельца.
`OWNER_IDO`	`string`	URL для получения информации о IDO владельца.
`OWNER_STAKE_FARMS`	`string`	URL для получения ферм, на которых владелец застейкался.
`OWNER_LOCK_POSITION`	`string`	URL для получения информации о заблокированных позициях владельца.

Конфигурация свопов:

Параметр	Тип	Описание
`SWAP_HOST`	`string`	URL для основного хоста свопов.
`SWAP_COMPUTE`	`string`	URL для вычисления свопа.
`SWAP_TX`	`string`	URL для отправки транзакции свопа.

Дополнительные параметры:

Параметр	Тип	Описание
`MINT_PRICE`	`string`	URL для получения цены минта токена.
`MIGRATE_CONFIG`	`string`	URL для получения конфигурации миграции.
`PRIORITY_FEE`	`string`	URL для получения информации о приоритетной комиссии.
`CPMM_LOCK`	`string`	URL для получения статуса блокировки CPMM.

Пример использования:

```
const urlConfigs = {
    BASE_HOST: 'https://api.raydium.io',
    FARM_ARP: 'https://api.raydium.io/farm_arp',
    SWAP_HOST: 'https://api.raydium.io/swap',
};
```

```
const raydium = await Raydium.load({
    connection,
    cluster: 'mainnet',
    owner,
    urlConfigs,
});
```

Метод: `parseTokenAccountResp`

Метод `parseTokenAccountResp` из Raydium SDK используется для преобразования данных токен-аккаунта в удобный формат. Позволяет объединить данные аккаунтов Solana и токен-аккаунтов (включая токены программы Token 2022) в единую структуру, которая содержит как обработанную, так и исходную информацию

Сигнатура метода:

```
const tokenAccountData = parseTokenAccountResp(options);
```

Имеет тип:

```
declare function parseTokenAccountResp({
    owner,
    solAccountResp,
    tokenAccountResp
}: ParseTokenAccount): {
    tokenAccounts: TokenAccount[];
    tokenAccountRawInfos: TokenAccountRaw[];
};
```

Параметры:

Метод принимает объект с тремя ключами:

Параметр	Тип	Обязательный	Описание
`owner`	`PublicKey`	Да	Публичный ключ владельца аккаунтов.
`solAccountResp`	`AccountInfo<Buffer>`	Да	Данные аккаунта Solana, полученные через

Параметр	Тип	Обязательный	Описание
			`getAccountInfo`.
`tokenAccountResp`	`{ context: Context, value: Token[] }`	Да	Данные токен-аккаунтов, полученные с помощью `getTokenAccountsByOwner`.

Возвращаемое значение:

Метод возвращает объект со следующими полями:

Поле	Тип	Описание
`tokenAccounts`	`TokenAccount[]`	Массив обработанных токен-аккаунтов с удобной структурой данных.
`tokenAccountRawInfos`	`TokenAccountRaw[]`	Массив сырых данных токен-аккаунтов (без обработки).

Пример использования:

```
export const fetchTokenAccountData = async () => {
  try {
    const [solAccountResp, tokenAccountResp, token2022Req] = await Promise.all([
      connection.getAccountInfo(owner.publicKey),
      connection.getTokenAccountsByOwner(owner.publicKey, {
        programId: TOKEN_PROGRAM_ID,
      }),
      connection.getTokenAccountsByOwner(owner.publicKey, {
        programId: TOKEN_2022_PROGRAM_ID,
      }),
    ])

    const combinedTokenAccounts = [
      ...tokenAccountResp.value,
      ...token2022Req.value,
    ]

    const tokenAccountData = parseTokenAccountResp({
      owner: owner.publicKey,
      solAccountResp,
      tokenAccountResp: {
        context: tokenAccountResp.context,
        value: combinedTokenAccounts,
```

```
      },
    })

    return tokenAccountData
  } catch (error) {
    console.error('Error in fetchTokenAccountData:', error)
    throw error
  }
}
```

Api methods

1. Метод: `raydium.api.getTokenList`

Метод `raydium.api.getTokenList` возвращает список токенов по умолчанию из Raydium API для основной сети (mainnet).

Сигнатура метода:

```
const data = await raydium.api.getTokenList();
```

Имеет тип:

```
getTokenList(): Promise<{
        mintList: ApiV3Token[];
        blacklist: string[];
        whiteList: string[];
    }>;

   declare type ApiV3Token = {
    chainId: number;
    address: string;
    programId: string;
    logoURI: string;
    symbol: string;
    name: string;
    decimals: number;
    tags: string[];
    extensions: ExtensionsItem;
    freezeAuthority?: string;
    mintAuthority?: string;
};
```

Описание:

Этот метод позволяет получить полный список токенов, поддерживаемых Raydium на основной сети Solana (mainnet). Список включает метаданные о токенах, такие как адреса, символы, количество десятичных знаков и другая информация.

Возвращаемое значение:

Метод возвращает объект с тремя основными полями:

Поле	Тип	Описание
`blacklist`	`string[]`	Массив публичных ключей токенов, находящихся в черном списке.
`mintList`	`Array<TokenInfo>`	Массив объектов, каждый из которых содержит информацию о конкретном токене.
`whiteList`	`string[]`	Массив публичных ключей токенов, находящихся в белом списке.

Структура объекта токена (`TokenInfo`):

Каждый элемент массива `mintList` представляет собой объект со следующими полями:

Поле	Тип	Описание
`chainId`	`number`	Идентификатор сети (например, 101 для mainnet).
`address`	`string`	Адрес токена в сети Solana.
`programId`	`string`	Адрес программы токена (например, стандартный SPL Token или Token 2022).
`logoURI`	`string`	URL к изображению логотипа токена.
`symbol`	`string`	Символ токена (например, `ETH` , `PYUSD`).
`name`	`string`	Полное название токена.
`decimals`	`number`	Количество знаков после запятой (дробная часть).
`tags`	`string[]`	Массив тегов, описывающих особенности токена (например, `hasFreeze`).
`extensions`	`object`	Дополнительные метаданные (например, ссылки на ресурсы или внешние данные).

Пример использования:

```
async function fetchTokenList() {
    try {
        const data = await raydium.api.getTokenList();
        console.log('Черный список токенов:', data.blacklist);
        console.log('Белый список токенов:', data.whiteList);

        data.mintList.forEach((token) => {
            console.log(`Токен: ${token.symbol}`);
            console.log(`Адрес: ${token.address}`);
            console.log(`Десятичные знаки: ${token.decimals}`);
            console.log(`Логотип: ${token.logoURI}`);
            console.log(`Программа: ${token.programId}`);
            console.log(`Название: ${token.name}`);
            console.log(`Теги: ${token.tags.join(', ')}`);
            console.log('---------------------------');
        });
    } catch (error) {
        console.error('Ошибка получения списка токенов:', error);
    }
}

fetchTokenList();
```

Пример возвращаемых данных:


```
{
  "blacklist": [],
  "mintList": [
    {
      "chainId": 101,
      "address": "2b1kV6DkPAnxd5ixfnxCpjxmKwqjjaYmCZfHsFu24GXo",
      "programId": "TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb",
      "logoURI": "https://img-
v1.raydium.io/icon/2b1kV6DkPAnxd5ixfnxCpjxmKwqjjaYmCZfHsFu24GXo.png",
      "symbol": "PYUSD",
      "name": "PayPal USD",
      "decimals": 6,
      "tags": ["hasFreeze"],
      "extensions": {}
    },
    {
      "chainId": 101,
      "address": "2FPyTwcZLUg1MDrwsyoP4D6s1tM7hAkHYRjkNb5w6Pxk",
      "programId": "TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA",
      "logoURI": "https://img-
```

```
v1.raydium.io/icon/2FPyTwcZLUg1MDrwsyoP4D6s1tM7hAkHYRjkNb5w6Pxk.png",
      "symbol": "ETH",
      "name": "Wrapped Ethereum (Sollet)",
      "decimals": 6,
      "tags": [],
      "extensions": {}
    }
  ],
  "whiteList": [
    "EPjFWdd5AufqSSqeM2qN1xzybapC8G4wEGGkZwyTDt1v",
    "Es9vMFrzaCERmJfrF4H2FYD4KCoNkY11McCe8BenwNYB"
  ]
}
```

- Возвращаемые данные актуальны только для основной сети (mainnet).
- Черный список (`blacklist`) обычно пуст, но может содержать токены, которые были запрещены на платформе.
- Белый список (`whiteList`) содержит проверенные токены, которые считаются безопасными для работы.

---

2. Метод: `raydium.api.getTokenInfo`

Метод `raydium.api.getTokenInfo` используется для получения информации о токенах по их адресам (монтам) в сети Solana. Этот метод позволяет получить метаданные токенов, распознаваемых Raydium, на основе их публичных ключей (монт).

Сигнатура метода:

```
const data = await raydium.api.getTokenInfo([
  "So11111111111111111111111111111111111111112",
  "4k3Dyjzvzp8eMZWUXbBCjEvwSkkk59S5iCNLY3QrkX6R",
]);
```

Имеет тип:

```
getTokenInfo(mint: (string | PublicKey)[]): Promise<ApiV3Token[]>;
// см. ApiV3Token в raydium.api.getTokenList
```

Описание:

Метод принимает массив строк или объектов `PublicKey`, содержащих адреса токенов (монт), и возвращает массив объектов с подробной информацией о каждом токене, который распознается Raydium.

Параметры:

Параметр	Тип	Обязательный	Описание
`mint`	`string[]` или `PublicKey[]`	Да	Массив публичных ключей токенов, информацию о которых нужно получить.

Возвращаемое значение:

Метод возвращает промис, который разрешается в массив объектов типа `ApiV3Token`.

Структура объекта токена (`ApiV3Token`):

Поле	Тип	Описание
`chainId`	`number`	Идентификатор сети (например, 101 для mainnet).
`address`	`string`	Адрес токена в сети Solana.
`programId`	`string`	Адрес программы токена (например, стандартный SPL Token).
`logoURI`	`string`	URL к логотипу токена.
`symbol`	`string`	Символ токена (например, `WSOL`, `RAY`).
`name`	`string`	Полное имя токена.
`decimals`	`number`	Количество знаков после запятой (дробная часть).
`tags`	`string[]`	Массив тегов, описывающих особенности токена.
`extensions`	`ExtensionsItem`	Дополнительные метаданные (например, данные о комиссии).
`freezeAuthority?`	`string` (опционально)	Адрес владельца прав заморозки токена (если есть).
`mintAuthority?`	`string` (опционально)	Адрес владельца прав на выпуск токена (если есть).

Структура объекта расширений (`ExtensionsItem`):

Поле	Тип	Описание
`coingeckoId?`	`string` (опционально)	Идентификатор токена на CoinGecko.
`feeConfig?`	`TransferFeeDataBaseType` (опционально)	Конфигурация комиссий при переводе.

Пример использования:

```
async function getTokenInformation() {
    try {
        const data = await raydium.api.getTokenInfo([
            "So11111111111111111111111111111111111111112",
            "4k3Dyjzvzp8eMZWUXbBCjEvwSkkk59S5iCNLY3QrkX6R"
        ]);

        data.forEach(token => {
            console.log(`Название: ${token.name}`);
            console.log(`Символ: ${token.symbol}`);
            console.log(`Адрес: ${token.address}`);
            console.log(`Количество знаков после запятой: ${token.decimals}`);
            console.log(`Логотип: ${token.logoURI}`);
            console.log(`Программа: ${token.programId}`);
            console.log(`Расширения: ${JSON.stringify(token.extensions)}`);
            if (token.freezeAuthority) {
                console.log(`Freeze Authority: ${token.freezeAuthority}`);
            }
            if (token.mintAuthority) {
                console.log(`Mint Authority: ${token.mintAuthority}`);
            }
            console.log('---------------------------');
        });
    } catch (error) {
        console.error('Ошибка получения информации о токенах:', error);
    }
}

getTokenInformation();
```

Пример возвращаемых данных:

```
[
  {
    "chainId": 101,
    "address": "So11111111111111111111111111111111111111112",
    "programId": "TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA",
    "logoURI": "https://img-
v1.raydium.io/icon/So11111111111111111111111111111111111111112.png",
    "symbol": "WSOL",
    "name": "Wrapped SOL",
    "decimals": 9,
    "tags": [],
    "extensions": {}
  },
  {
    "chainId": 101,
    "address": "4k3Dyjzvzp8eMZWUXbBCjEvwSkkk59S5iCNLY3QrkX6R",
    "programId": "TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA",
    "logoURI": "https://img-
v1.raydium.io/icon/4k3Dyjzvzp8eMZWUXbBCjEvwSkkk59S5iCNLY3QrkX6R.png",
    "symbol": "RAY",
    "name": "Raydium",
    "decimals": 6,
    "tags": [],
    "extensions": {}
  }
]
```


- Поддерживает получение информации только для токенов, распознаваемых Raydium.
- Работает только на основной сети (mainnet).
- Удобно использовать для получения базовой информации о токенах перед операциями свопа или пулов ликвидности.

---

3. Метод: `raydium.api.getPoolList`

Метод `raydium.api.getPoolList` используется для получения списка пулов ликвидности на платформе Raydium в основной сети (mainnet). Этот метод позволяет получить информацию как о стандартных пулах, так и о пулах с концентрированной ликвидностью.

Сигнатура метода:

```
const data = await raydium.api.getPoolList({});
```

Имеет тип:

```
const data = await raydium.api.getPoolList(props?: FetchPoolParams):
Promise<PoolsApiReturn>;
```

Описание:

Метод возвращает список пулов в формате пагинации, с возможностью указания фильтров через параметры запроса.

Параметры (FetchPoolParams):

Параметр	Тип	Описание
`poolType`	`string`	Тип пула: `all`, `concentrated`, `standard`, `allFarm`, `concentratedFarm`, `standardFarm`.
`poolSortField`	`string`	Поле сортировки: `liquidity`, `volume_24h`, `7d`, `30d`, `fee_24h`, `7d`, `30d`, `apr_24h`, `7d`, `30d`.
`sortType`	`string`	Тип сортировки: `asc` или `desc`.
`page`	`number`	Номер страницы (по умолчанию 1).
`pageSize`	`number`	Количество записей на странице (по умолчанию 20).
`ids`	`string[]`	Массив ID пулов для выборочного поиска.

Возвращаемое значение:

Метод возвращает промис, который разрешается в объект типа `PoolsApiReturn`.

Структура возвращаемого объекта (`PoolsApiReturn`):

Поле	Тип	Описание
`count`	`number`	Общее количество пулов.
`hasNextPage`	`boolean`	Есть ли следующая страница с данными.
`data`	`ApiV3PoolInfoItem[]`	Массив объектов, содержащих информацию о пулах.

Структура объекта пула (`ApiV3PoolInfoStandardItemCpmm`):

Поле	Тип	Описание
`type`	`"Standard"`	Тип пула (в данном случае стандартный).
`lpMint`	`ApiV3Token`	Объект токена LP.
`lpPrice`	`number`	Цена LP-токена.
`lpAmount`	`number`	Объем LP-токена в пуле.
`config`	`ApiCpmmConfigV3`	Конфигурация пула.

Структура объекта конфигурации пула (`ApiCpmmConfigV3`):

Поле	Тип	Описание
`id`	`string`	Идентификатор пула.
`index`	`number`	Индекс пула.
`protocolFeeRate`	`number`	Комиссия протокола.
`tradeFeeRate`	`number`	Комиссия за трейд.
`fundFeeRate`	`number`	Комиссия за управление средствами.
`createPoolFee`	`string`	Комиссия за создание пула.

Пример использования:

```
async function fetchPoolList() {
    try {
        const data = await raydium.api.getPoolList({
            poolType: 'standard',
            poolSortField: 'liquidity',
            sortType: 'desc',
            page: 1,
            pageSize: 10,
        });

        console.log(`Общее количество пулов: ${data.count}`);
        console.log(`Есть ли следующая страница: ${data.hasNextPage}`);

        data.data.forEach((pool) => {
            console.log(`Пул ID: ${pool.config.id}`);
            console.log(`Тип: ${pool.type}`);
            console.log(`Цена LP: ${pool.lpPrice}`);
            console.log(`Объем LP: ${pool.lpAmount}`);
```

```
            console.log(`Комиссия протокола: ${pool.config.protocolFeeRate}`);
            console.log(`Комиссия за трейд: ${pool.config.tradeFeeRate}`);
            console.log(`Комиссия за создание пула: ${pool.config.createPoolFee}`);
            console.log('--------------------------');
        });
    } catch (error) {
        console.error('Ошибка получения списка пулов:', error);
    }
}

fetchPoolList();
```

Пример возвращаемых данных:


```
{
  "count": 0,
  "data": [],
  "hasNextPage": true
}
```


Пример объекта пула:


```
{
  "type": "Standard",
  "lpMint": {
    "chainId": 101,
    "address": "So11111111111111111111111111111111111111112",
    "programId": "TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA",
    "logoURI": "https://img-
v1.raydium.io/icon/So11111111111111111111111111111111111111112.png",
    "symbol": "WSOL",
    "name": "Wrapped SOL",
    "decimals": 9,
    "tags": [],
    "extensions": {}
  },
  "lpPrice": 1.2345,
  "lpAmount": 1000,
  "config": {
    "id": "pool123",
    "index": 1,
    "protocolFeeRate": 0.003,
    "tradeFeeRate": 0.001,
    "fundFeeRate": 0.0005,
```

```
    "createPoolFee": "0.02"
  }
}
```

---

4. Метод: `raydium.api.fetchPoolById`

Метод `raydium.api.fetchPoolById` используется для получения информации о конкретных пулах ликвидности на платформе Raydium на основе их идентификаторов (ID). Работает только в основной сети (mainnet).

Сигнатура метода:

```
// ids: join pool ids by comma(,)
const data = await raydium.api.fetchPoolById({
  ids:
"AVs9TA4nWDzfPJE9gGVNJMVhcQy3V9PGazuz33BfG2RA,8sLbNZoA1cfnvMJLPfp98ZLAnFSYCFApfJKMb
iXNLwxj",
});
```

Имеет тип:

```
const data = await raydium.api.fetchPoolById(props: { ids: string }):
Promise<ApiV3PoolInfoItem[]>;
```

Описание:

Метод принимает объект с параметром `ids`, представляющим собой строку, содержащую ID пулов, разделённых запятой. Возвращает массив объектов, каждый из которых содержит полную информацию о пуле.

Параметры:

Параметр	Тип	Обязательный	Описание
`ids`	`string`	Да	Строка с ID пулов, разделённых запятой.

Пример параметра:

```
{
  ids:
"AVs9TA4nWDzfPJE9gGVNJMVhcQy3V9PGazuz33BfG2RA,8sLbNZoA1cfnvMJLPfp98ZLAnFSYCFApfJKMb
iXNLwxj"
}
```

Возвращаемое значение:

Метод возвращает промис, который разрешается в массив объектов типа `ApiV3PoolInfoItem`.

Структура объекта пула (`ApiV3PoolInfoItem`):

Поле	Тип	Описание
`type`	`string`	Тип пула: `Standard`, `Concentrated`.
`programId`	`string`	Программа, связанная с пулом.
`id`	`string`	Идентификатор пула.
`mintA`, `mintB`	`ApiV3Token`	Объекты токенов, участвующих в пуле.
`price`	`number`	Текущая цена пары токенов в пуле.
`mintAmountA`, `mintAmountB`	`number`	Количество токенов A и B в пуле.
`feeRate`	`number`	Комиссия за транзакцию.
`tvl`	`number`	Общая заблокированная стоимость (TVL) в пуле.
`lpPrice`	`number`	Цена LP-токена.
`lpAmount`	`number`	Количество LP-токенов в пуле.
`burnPercent`	`number`	Процент сжигания токенов при удалении ликвидности.
`day`, `week`, `month`	`ApiV3PoolStats`	Статистика пула за день, неделю и месяц.
`farmOngoingCount`	`number`	Количество активных фарминг-программ.
`farmFinishedCount`	`number`	Количество завершённых фарминг-программ.
`rewardDefaultPoolInfos`	`string`	Имя пула наград.
`rewardDefaultInfos`	`ApiV3RewardInfo[]`	Информация о наградах.

Структура статистики пула (`ApiV3PoolStats`):

Поле	Тип	Описание
`volume`	`number`	Объём торгов за период.
`volumeQuote`	`number`	Объём в котируемом токене.
`volumeFee`	`number`	Комиссия за период.
`apr`	`number`	Годовая доходность (APR).
`feeApr`	`number`	Доходность от комиссий.
`rewardApr`	`number[]`	Доходность от наград.

Пример использования:

```
async function fetchPoolInfo() {
    try {
        const data = await raydium.api.fetchPoolById({
            ids:
"AVs9TA4nWDzfPJE9gGVNJMVhcQy3V9PGazuz33BfG2RA,8sLbNZoA1cfnvMJLPfp98ZLAnFSYCFApfJKMb
iXNLwxj"
        });

        data.forEach(pool => {
            console.log(`Пул ID: ${pool.id}`);
            console.log(`Тип: ${pool.type}`);
            console.log(`Цена: ${pool.price}`);
            console.log(`Количество токена A: ${pool.mintAmountA}`);
            console.log(`Количество токена B: ${pool.mintAmountB}`);
            console.log(`TVL: ${pool.tvl}`);
            console.log(`Комиссия: ${pool.feeRate}`);
            console.log(`Цена LP: ${pool.lpPrice}`);
            console.log(`Количество LP: ${pool.lpAmount}`);
            console.log(`Процент сжигания: ${pool.burnPercent}`);
            console.log('Статистика за день:', pool.day);
            console.log('Статистика за неделю:', pool.week);
            console.log('Статистика за месяц:', pool.month);
            console.log('----------------------------');
        });
    } catch (error) {
        console.error('Ошибка получения информации о пуле:', error);
    }
}
```

```
fetchPoolInfo();
```

Пример возвращаемых данных:


```
[
  {
    "type": "Standard",
    "programId": "675kPX9MHTjS2zt1qfr1NYHuzeLXfQM9H24wFSUt1Mp8",
    "id": "AVs9TA4nWDzfPJE9gGVNJMVhcQy3V9PGazuz33BfG2RA",
    "mintA": {
      "address": "4k3Dyjzvzp8eMZWUXbBCjEvwSkkk59S5iCNLY3QrkX6R",
      "symbol": "RAY",
      "name": "Raydium"
    },
    "mintB": {
      "address": "So11111111111111111111111111111111111111112",
      "symbol": "WSOL",
      "name": "Wrapped SOL"
    },
    "price": 0.0148,
    "mintAmountA": 1630979.96,
    "mintAmountB": 24183.84,
    "feeRate": 0.0025,
    "tvl": 5558341.54,
    "day": {
      "volume": 1920098.52,
      "apr": 31.52
    },
    "lpPrice": 12.54,
    "lpAmount": 443224.97,
    "burnPercent": 0.08
  }
]
```


Данные включают объёмы торгов, доходность и данные по LP-токенам.

---

5. Метод: `raydium.api.fetchPoolByMints`

Метод `fetchPoolByMints` используется для получения списка пулов по заданным токенам (mint1 и mint2) на основной сети (mainnet) с использованием Raydium API. Метод позволяет искать пулы на основе двух токенов (адресов) и дополнительных параметров.

Сигнатура метода:

```
const data = await raydium.api.fetchPoolByMints({
  mint1: "So11111111111111111111111111111111111111112",
  mint2: "4k3Dyjzvzp8eMZWUXbBCjEvwSkkk59S5iCNLY3QrkX6R", // optional,
  // extra params: https://github.com/raydium-io/raydium-sdk-
V2/blob/master/src/api/type.ts#L249
});
```

Имеет тип:

```
const data = await raydium.api.fetchPoolByMints({
    mint1: string | PublicKey;
    mint2?: string | PublicKey;
} & Omit<FetchPoolParams, "pageSize">): Promise<PoolsApiReturn>;
```

Аргументы

Параметр	Тип	Обязательный	Описание
mint1	string | PublicKey	Да	Адрес первого токена.
mint2	string | PublicKey	Нет	Адрес второго токена.
type	PoolFetchType	Нет	Тип пула: `all`, `standard`, `concentrated`, `allFarm`, и т.д.
sort	string	Нет	Критерий сортировки: `liquidity`, `volume24h`, `apr7d`, и т.д.
order	string	Нет	Порядок сортировки: `asc` или `desc`.
page	number	Нет	Номер страницы для пагинации.

Типы пулов (PoolFetchType)

Значение	Описание
`All`	Все пулы
`Standard`	Стандартные пулы
`Concentrated`	Концентрированные пулы
`AllFarm`	Все фермы

Значение	Описание
`StandardFarm`	Стандартные фермы
`ConcentratedFarm`	Концентрированные фермы

Пример использования

```
const data = await raydium.api.fetchPoolByMints({
  mint1: "So11111111111111111111111111111111111111112",
  mint2: "4k3Dyjzvzp8eMZWUXbBCjEvwSkkk59S5iCNLY3QrkX6R",
  type: "standard",
  sort: "volume24h",
  order: "desc"
});
```

Возвращаемое значение метода `fetchPoolByMints`

Метод возвращает объект, содержащий информацию о пулах на основе заданных токенов. Ниже приведена структура возвращаемого объекта.

Общая структура ответа

Поле	Тип	Описание
count	number	Общее количество пулов, соответствующих запросу.
data	массив	Массив объектов пулов с детализированной информацией.
hasNextPage	boolean	Указывает, есть ли следующая страница данных.

Структура объекта пула

Поле	Тип	Описание
type	string	Тип пула (`Concentrated` , `Standard`).
programId	string	Идентификатор программы пула.
id	string	Уникальный идентификатор пула.
mintA	объект	Информация о первом токене пула.

Поле	Тип	Описание
mintB	объект	Информация о втором токене пула.
rewardDefaultPoolInfos	string	Тип пула вознаграждений (например, "Clmm").
rewardDefaultInfos	массив	Список вознаграждений по пулу.
price	number	Текущая цена токена A относительно B.
mintAmountA	number	Количество токена A в пуле.
mintAmountB	number	Количество токена B в пуле.
feeRate	number	Комиссия за транзакцию в пуле.
openTime	string	Время открытия пула в формате Unix.
tvl	number	Общая заблокированная стоимость пула.
day	объект	Метрики пула за последний день.
week	объект	Метрики пула за последнюю неделю.
month	объект	Метрики пула за последний месяц.
pooltype	массив	Типы пулов (например, фермы).
farmUpcomingCount	number	Количество предстоящих ферм.
farmOngoingCount	number	Количество активных ферм.
farmFinishedCount	number	Количество завершенных ферм.
config	объект	Конфигурация пула.
burnPercent	number	Процент сжигания токенов.

Структура объекта токена (mintA, mintB)

Поле	Тип	Описание
chainId	number	Идентификатор сети (например, 101).
address	string	Адрес токена в сети.
programId	string	Идентификатор программы токена.
logoURI	string	Ссылка на изображение токена.
symbol	string	Символьное обозначение токена (например, `WSOL`).
name	string	Название токена (например, `Wrapped SOL`).
decimals	number	Количество десятичных знаков у токена.
tags	массив	Дополнительные метки токена.

Поле	Тип	Описание
extensions	объект	Дополнительные расширенные данные.

Структура объекта вознаграждений (rewardDefaultInfos)

Поле	Тип	Описание
mint	объект	Информация о токене вознаграждения.
perSecond	string	Количество вознаграждений в секунду.
startTime	string	Время начала начисления вознаграждений (Unix).
endTime	string	Время окончания начисления вознаграждений (Unix).

Структура метрик за день, неделю и месяц (day, week, month)

Поле	Тип	Описание
volume	number	Общий объем торгов за указанный период.
volumeQuote	number	Объем торгов в эквиваленте котируемого токена.
volumeFee	number	Сумма комиссии за операции в пуле за указанный период.
apr	number	Годовая доходность пула.
feeApr	number	Годовая доходность от комиссий.
priceMin	number	Минимальная цена токена A относительно B за период.
priceMax	number	Максимальная цена токена A относительно B за период.
rewardApr	массив	Годовая доходность от вознаграждений.

Структура объекта конфигурации пула (config)

Поле	Тип	Описание
id	string	Идентификатор конфигурации пула.

Поле	Тип	Описание
index	number	Индекс конфигурации.
protocolFeeRate	number	Комиссия протокола в базисных пунктах.
tradeFeeRate	number	Торговая комиссия в базисных пунктах.
tickSpacing	number	Шаг изменения цены в пуле.
fundFeeRate	number	Комиссия на фонд пула в базисных пунктах.
defaultRange	number	Стандартный диапазон изменения цены.
defaultRangePoint	массив	Массив значений диапазона изменения цены.

Пример ответа


```
{
  "count": 29,
  "data": [
        {
      "type": "Concentrated",
      "programId": "CAMMCzo5YL8w4VFF8KVHrK22GGUsp5VTaW7grrKgrWqK",
      "id": "2AXXcN6oN9bBT5owwmTH53C7QHUXvhLeu718Kqt8rvY2",
      "mintA": {
        "chainId": 101,
        "address": "So11111111111111111111111111111111111111112",
        "programId": "TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA",
        "logoURI": "https://img-
v1.raydium.io/icon/So11111111111111111111111111111111111111112.png",
        "symbol": "WSOL",
        "name": "Wrapped SOL",
        "decimals": 9,
        "tags": [],
        "extensions": {}
      },
      "mintB": {
        "chainId": 101,
        "address": "4k3Dyjzvzp8eMZWUXbBCjEvwSkkk59S5iCNLY3QrkX6R",
        "programId": "TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA",
        "logoURI": "https://img-
v1.raydium.io/icon/4k3Dyjzvzp8eMZWUXbBCjEvwSkkk59S5iCNLY3QrkX6R.png",
        "symbol": "RAY",
        "name": "Raydium",
        "decimals": 6,
        "tags": [],
        "extensions": {}
```

```
      },
      "rewardDefaultPoolInfos": "Clmm",
      "rewardDefaultInfos": [
        {
          "mint": {
            "chainId": 101,
            "address": "4k3Dyjzvzp8eMZWUXbBCjEvwSkkk59S5iCNLY3QrkX6R",
            "programId": "TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA",
            "logoURI": "https://img-
v1.raydium.io/icon/4k3Dyjzvzp8eMZWUXbBCjEvwSkkk59S5iCNLY3QrkX6R.png",
            "symbol": "RAY",
            "name": "Raydium",
            "decimals": 6,
            "tags": [],
            "extensions": {}
          },
          "perSecond": "992",
          "startTime": "1740092400",
          "endTime": "1746741600"
        }
      ],
      "price": 66.29455737789726,
      "mintAmountA": 10363.532505589,
      "mintAmountB": 950650.214292,
      "feeRate": 0.0005,
      "openTime": "0",
      "tvl": 2836035.75,
      "day": {
        "volume": 13151723.215064714,
        "volumeQuote": 7113886.228569916,
        "volumeFee": 6575.861607532359,
        "apr": 86.53609885132438,
        "feeApr": 84.63,
        "priceMin": 59.85565991960619,
        "priceMax": 72.08902404143883,
        "rewardApr": [
          1.906098851324388
        ]
      },
      "week": {
        "volume": 71534717.7838236,
        "volumeQuote": 39377408.84204179,
        "volumeFee": 35767.358891911805,
        "apr": 39.74609885132439,
        "feeApr": 37.84,
        "priceMin": 59.85565991960619,
```

```
        "priceMax": 74.7762483322983,
        "rewardApr": [
          1.906098851324388
        ]
      },
      "month": {
        "volume": 230972700.9253229,
        "volumeQuote": 127554468.780804,
        "volumeFee": 115486.35046266147,
        "apr": 50.776098851324384,
        "feeApr": 48.87,
        "priceMin": 59.85565991960619,
        "priceMax": 83.96128948401767,
        "rewardApr": [
          1.906098851324388
        ]
      },
      "pooltype": [],
      "farmUpcomingCount": 0,
      "farmOngoingCount": 1,
      "farmFinishedCount": 0,
      "config": {
        "id": "HfERMT5DRA6C1TAqecrJQFpmkf3wsWTMncqnj3RDg5aw",
        "index": 2,
        "protocolFeeRate": 120000,
        "tradeFeeRate": 500,
        "tickSpacing": 10,
        "fundFeeRate": 40000,
        "defaultRange": 0.1,
        "defaultRangePoint": [
          0.01,
          0.05,
          0.1,
          0.2,
          0.5
        ]
      },
      "burnPercent": 0.17
    },
  ],
  "hasNextPage": false
}
```

6. Метод: `raydium.api.fetchFarmInfoById`

Описание

Метод `fetchFarmInfoById` используется для получения информации о ферме (фарминге) по заданным идентификаторам (IDs) на основной сети (mainnet) с использованием Raydium API. Метод принимает строку с перечислением ID ферм через запятую и возвращает массив с детализированной информацией о каждой ферме.

Синтаксис:

```
const data = await raydium.api.fetchFarmInfoById({
    ids: string;
}): Promise<FormatFarmInfoOut[]>;
```

Аргументы

Поле	Тип	Обязательный	Описание
ids	string	Да	Строка с ID ферм, разделенными запятой.

Пример использования

```
const data = await raydium.api.fetchFarmInfoById({
  ids:
"4EwbZo8BZXP5313z5A2H11MRBP15M5n6YxfmkjXESKAW,HUDr9BDaAGqi37xbQHzxCyXvfMCKPTPNF8g9c
9bPu1Fu",
});
```

Возвращаемое значение

Метод возвращает массив объектов типа `FormatFarmInfoOut`, содержащих следующую информацию:

Структура объекта фермы

Поле	Тип	Описание
programId	string	Идентификатор программы фарминга.
id	string	Уникальный идентификатор фермы.

Поле	Тип	Описание
symbolMints	массив	Массив объектов с информацией о токенах фермы.
lpMint	объект	Объект с данными об LP-токене фермы.
tvl	number	Общая заблокированная стоимость фермы.
lpPrice	number	Цена LP-токена.
apr	number	Годовая доходность (APR) фермы.
tags	массив	Массив тегов, характеризующих ферму (например, "Stake").
rewardInfos	массив	Массив объектов с информацией о вознаграждениях в ферме.

Структура объекта токена (symbolMints, lpMint)

Поле	Тип	Описание
chainId	number	Идентификатор сети (например, 101).
address	string	Адрес токена в сети.
programId	string	Идентификатор программы токена.
logoURI	string	Ссылка на изображение токена.
symbol	string	Символьное обозначение токена (например, `RAY`).
name	string	Название токена (например, `Raydium`).
decimals	number	Количество десятичных знаков у токена.
tags	массив	Дополнительные метки токена.
extensions	объект	Дополнительные расширенные данные.

Структура объекта вознаграждений (rewardInfos)

Поле	Тип	Описание
mint	объект	Информация о токене вознаграждения.
type	string	Тип вознаграждения (например, "Standard SPL").
apr	number	Годовая доходность от вознаграждения.
perSecond	string	Количество вознаграждений в секунду.

Пример полного ответа


```
[
  {
    "programId": "EhhTKczWMGQt46ynNeRX1WfeagwwJd7ufHvCDjRxjo5Q",
    "id": "4EwbZo8BZXP5313z5A2H11MRBP15M5n6YxfmkjXESKAW",
    "symbolMints": [
      {
        "chainId": 101,
        "address": "4k3Dyjzvzp8eMZWUXbBCjEvwSkkk59S5iCNLY3QrkX6R",
        "programId": "TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA",
        "logoURI": "https://img-
v1.raydium.io/icon/4k3Dyjzvzp8eMZWUXbBCjEvwSkkk59S5iCNLY3QrkX6R.png",
        "symbol": "RAY",
        "name": "Raydium",
        "decimals": 6
      }
    ],
    "lpMint": {
      "chainId": 101,
      "address": "4k3Dyjzvzp8eMZWUXbBCjEvwSkkk59S5iCNLY3QrkX6R",
      "programId": "TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA",
      "logoURI": "https://img-
v1.raydium.io/icon/4k3Dyjzvzp8eMZWUXbBCjEvwSkkk59S5iCNLY3QrkX6R.png",
      "symbol": "RAY",
      "name": "Raydium",
      "decimals": 6
    },
    "tvl": 56119674.85744179,
    "lpPrice": 1.7187496458382656,
    "apr": 0.0628,
    "tags": ["Stake"],
    "rewardInfos": [
      {
        "mint": {
          "chainId": 101,
          "address": "4k3Dyjzvzp8eMZWUXbBCjEvwSkkk59S5iCNLY3QrkX6R",
          "programId": "TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA",
          "logoURI": "https://img-
v1.raydium.io/icon/4k3Dyjzvzp8eMZWUXbBCjEvwSkkk59S5iCNLY3QrkX6R.png",
          "symbol": "RAY",
          "name": "Raydium",
          "decimals": 6
        },
        "type": "Standard SPL",
        "apr": 0.0628,
```

```
        "perSecond": "65280"
      }
    ]
  }
]
```