# How to use tonana data transfer (RU)

## NEAR → TON

NEAR Source chain. Для отправки транзакции с данными:

Для перевода данных из NEAR в TON надо вызвать функцию payToWallet на контракте tonana.near. Первый параметр - target. Тут надо указать tonanawallet.near (валет бриджа с которого будет отслеживаться перевод). Второй параметр - message. Если вы хотите совершить перевод и положить данные то вначале строки поля message идет “SM#” как флаг. Потом добавляем “TON#” - это сеть назначения. Потом добавляем адрес на сети назначения (на тоне) и добавляем знак “#” после чего идет строка данных. По сути нету никаких ограничений по формату передоваемых данных - можете положить JSON / можете положить HEX или числа. Самое интересное - можете положить HASH данных и передать только его а сами данные передать незащищенным способом (просто через FTP например) и на таргет чеине уже сверить HASH данных с хешем пришедшего файла. Тут нету никаких ограничений.

Вот пример создания транзакции из нашего опенсорс репозитория:

```
await window.contract.account._signAndSendTransaction({
  receiverId: process.env.REACT_APP_NEAR_CONTRACT,
  actions: [
    transactions.functionCall(
      'payToWallet',
      {
        target: process.env.REACT_APP_BACK_NEAR_WALLET,
        message: `${openData ? "SM#" : ""}${netTo}#${openData? add : walletTo}${openData ? `#${btoa(params)}` : ""}`
      },
      new TonWeb.utils.BN(40000000000000),
      new TonWeb.utils.BN(utils.format.parseNearAmount(amount)+'')
    )
  ],
  walletCallbackUrl: 'https://app.tonana.org/?isnear=true'
})
```

После того как юзер подпишет транзакцию ваше приложение должно оповестить оракл тонаны о том, что транзакция была проведена. Поэтому вам надо сделать fetch запрос на наш API endpoint (https://api.tonana.org/). В боди которого кладется два параметра - hash транзакции и sourceChain (в данном случае это будет “near”).

Вот пример оповещения оракла тонаны о том что запрос был произведен.

```
if (transactionHashes) {
  fetch(process.env.REACT_APP_STATE === "dev" ? "http://localhost:8092" : process.env.REACT_APP_STATE === "dev-remote" ? "https://dev.api.tonana.org"
: "https://api.tonana.org/", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({
      hash: transactionHashes,
      sourceChain: "near",
    }),
  })
    .then((e) => e.text())
```

Исходный файл где можно найти пример нир транзакции: https://github.com/tonanadao/tonana-swap-v1-fe/blob/main/src/logic/transaction/MakeNEARTrx.ts

TON Target chain. Для приема данных:

После того как вы отпраили транзакцию на нире оракл тонаны ее провалидирует и если с ней все хорошо, то он отправит транзакцию на тоне. Внутри транзакции будут закодированы ваши данные. Чтобы получить их надо их правильно вытащить.

Берем транзакцию которая пришла на ваш кошелек (this.transaction - это полученная нужная транакция с RPC провайдера). Берем msg_data.body - это сообщение транзакции. После чего надо правильно все декодировать. Пример на скриншоте.

```
get_source_transaction_msg() {
  return decodeOffChainContent(
    Cell.fromBoc(
      Buffer.from(
        TonWeb.utils.base64ToBytes(this.transaction.in_msg.msg_data.body)
      )
    )[0]
  );
}
```

Где функция decodeOffChainContent разбирает дерево клеток по нужному префиксу. Ваши данные лежат в дереве клеток кусками до 127байт в одной клетке. Поэтому вам надо правильно достать их и “склеить” обратно - для этого есть функция хелпер decodeOffChainContent.

Линк на файл с функцией хелпера - https://github.com/tonanadao/tonana-swap-v1-fe/blob/main/src/logic/transaction/BOCcontent.ts

После этого у вас на руках будет расшифрованное сообщение как результат работы функции. Важно отметить что результат - это строка вида “SM#TON#WALLET#DATA”. Дальше можете делать с ней что хотите - это данные уже на тоне которые пришли вам с нира.

## TON → NEAR

TON Source chain. Для отправки транзакции с данными:

Для перевода данных из TON в NEAR надо отправить правильно собранную транзакцию. На наш тон валет (EQCyUNjJ6-BJqqhtsiVRn1W4tPAT4jyCDT9DdyRNT_QwrWs9). Если вы хотите совершить перевод и положить данные то для начала вам надо собрать строку в которой сначала идет “SM#” как флаг. Потом добавляем “NEAR#” - это сеть назначения. Потом добавляем адрес на сети назначения (на нире) и добавляем знак “#” после чего идет строка данных. По сути нету никаких ограничений по формату передоваемых данных - можете положить JSON / можете положить HEX или числа. Самое интересное - можете положить HASH данных и передать только его а сами данные передать незащищенным способом (просто через FTP например) и на таргет чеине уже сверить HASH данных с хешем пришедшего файла. Тут нету никаких ограничений. После чего полученная строка должна быть правильным

образом закодирована и положена как data трансфера. Ваши данные должны лежать в дереве клеток кусками до 127байт в одной клетке. Для этого есть функция-хелпер encodeOffChainContent (ссылка на нее https://github.com/tonanadao/tonana-swap-v1-fe/blob/main/src/logic/transaction/BOCcontent.ts). Полученные данные надо перегнать в base64 (конкретно в нашем случае для отправки транзакции на подпись tonweb валетом юзера)

Вот пример создания транзакции из нашего опенсорс репозитория:

```
await ton.send("ton_sendTransaction", [
        {
                to: process.env.REACT_APP_BACK_TON_WALLET,
                value: TonWeb.utils.toNano(Number(TONAmount)).toString(),
                data: encodeOffChainContent(
                        `${openData ? "SM#" : ""}${netTo}#${openData ? add : walletTo}${
                                openData ? `#${btoa(params)}` : ""
                        }`
                )
                        .toBoc()
                        .toString("base64"),
                dataType: "boc",
        },
])
```

После того как юзер подпишет транзакцию ваше приложение должно оповестить оракл тонаны о том, что транзакция была проведена. Поэтому вам надо сделать fetch запрос на наш API endpoint (https://api.tonana.org/). В боди которого кладется два параметра - hash транзакции и sourceChain (в данном случае это будет “ton”).

Вот пример оповещения оракла тонаны о том что запрос был произведен.

```
fetch(
        process.env.REACT_APP_STATE === "dev"
                ? "http://localhost:8092"
                : process.env.REACT_APP_STATE === "dev-remote"
                ? "https://dev.api.tonana.org"
                : "https://api.tonana.org/",
        {
                method: "POST",
                headers: { "Content-Type": "application/json" },
                body: JSON.stringify({
                        hash: data[0].transaction_id.hash,
                        sourceChain: "ton",
                }),
        }
```

NEAR Target chain. Для приема данных:

После того как вы отпраили транзакцию на тоне оракл тонаны ее провалидирует и если с ней все хорошо, то он отправит транзакцию на нире. Внутри транзакции будут закодированы ваши данные. Чтобы получить их надо их правильно вытащить.

Берем транзакцию которая пришла на наш прокси-контракт tonana.near (this.transaction - это полученная нужная транакция с RPC провайдера). Эта прокси транзакция сразу же была переслана вам на ваш кошелек. Но уже без данных (просто ниры). Все данные будут находиться именно в входящей транзакции на tonana.near. Вытаскиваем ее и берем поле message.

```
get_source_transaction_msg() {
  return JSON.parse(atob(this.transaction.transaction.actions[0].FunctionCall.args)).message
}
```

После этого у вас на руках будет расшифрованное сообщение как результат работы нашего бриджа. Важно отметить что результат - это строка вида “SM#NEAR#WALLET#DATA”. Дальше можете делать с ней что хотите - это данные уже на тоне которые пришли вам с TOHA.