How to use tonana data transfer (ENG) (deepL translated)

NEAR → TON

NEAR Source chain. To send a transaction with data:

To transfer data from NEAR to TON, you must call the payToWallet function on the tonana.near contract. The first parameter is target. Here you should specify tonanawallet.near (the bridge jack from which the transfer will be tracked). The second parameter is message. If you want to make a transfer and put the data in, you have to add "SM#" as a flag at the beginning of the message line. Then add "TON#" which is the destination network. Then add the address on the destination network (on the tone) and add a "#" sign after which the data string comes. In fact, there are no restrictions on the format of the transmitted data - you can put JSON / you can put HEX or numbers. The most interesting - you can put HASH data and send it only, and the data to pass unprotected way (just via FTP for example) and target check already HASH data with hash of the incoming file. There are no limitations here.

Here is an example of how to create a transaction from our Open Source repository:

```
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'
})
```

After the user signs the transaction, your application has to notify oracle tones that the transaction has been done. So you have to make a fetch request to our API endpoint (

https://api.tonana.org/

). In the body of which you put two parameters - transaction hash and sourceChain (in this case it will be "near").

Here is an example of notification by oracle tonana that the request was made.

```
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())
```

The source file where you can find an example nir transaction:

https://github.com/tonanadao/tonana-swap-v1-fe/blob/main/src/logic/transaction/MakeNEARTrx.ts

TON Target chain. To receive data:

After you send a transaction on a nir, oracle tones it down and if it's okay, it will send a transaction on a ton. Inside the transaction your data will be encoded. To get it you have to pull it out correctly.

Take the transaction that came to your wallet (this.transaction is the received needed transaction from the RPC provider). We take msg_data.body which is the message of the transaction. Then we have to decode it correctly. You can see an example in the screenshot.

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

Where the decodeOffChainContent function parses the cell tree by the desired prefix. Your data lies in the cell tree in chunks of up to 127 bytes per cell. So you need to get them correctly and "glue" them back together - that's what the decodeOffChainContent helper function is for.

Here is a link to the helper function:

https://github.com/tonanadao/tonana-swap-v1-fe/blob/main/src/logic/transaction/BOCcontent.ts

After that you will get a decoded message as a result of this function. It is important to note that result is string "SM#TON#WALLET#DATA". From here you can do what you want with it - it is already data on tone that came to you from nir.

TON → NEAR

TON Source chain. To send a transaction with data:

You have to send a properly assembled transaction to translate the data from TON to NEAR. Our tone is jack (EQCyUNjJ6-BJqqhtsiVRn1W4tPAT4jyCDT9DdyRNT_QwrWs9). If you want to translate and put data, first you have to assemble a line with "SM#" as a flag. Then add "NEAR#" which is the destination network. Then you add the address on the destination network (on the nir) and add a "#" sign after which the data string comes. In fact, there are no restrictions on the format of the transmitted data - you can put JSON / you can put HEX or numbers. The most interesting - you can put HASH data and send it only, and the data to pass unprotected way (just via FTP for example) and target check already HASH data with hash of the incoming file. There are no restrictions here. After that the received string should be encoded correctly and put as data transfer. Your data must lie in the cell tree in chunks of up to 127 bytes in one cell. There is a helper function encodeOffChainContent for this (link to it

https://github.com/tonanadao/tonana-swap-v1-fe/blob/main/src/logic/transaction/BOCcontent.ts

). The received data should be converted into base64 (in our case, to send a transaction to the user's tonweb signature)

Here is an example of transaction creation from our OpenSource repository:

```
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",
        },
])
```

After the user signs the transaction, your application must notify oracle tones that the transaction has been executed. So you have to make a fetch request to our API endpoint (

https://api.tonana.org/

). In the body of which you put two parameters - transaction hash and sourceChain (in this case it will be "ton").

Here is an example of oracle notifying tonana that the request was made.

```
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. To receive data:

After you send a transaction on the tone, the oracle tones it down and if all is well with it, it will send the transaction on the nir. Inside the transaction your data will be encoded. To get it you have to pull it out correctly.

We take the transaction that came to our tonana.near proxy (this.transaction is the received needed transaction from the RPC provider). This proxy transaction was immediately forwarded to your wallet. But without data (just nirs). All the data will be in the incoming transaction on tonana.near. Pull it out and take the message field.

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

After that you'll get a decrypted message as a result of our bridge. It's important to note, that the result is a string like "SM#NEAR#WALLET#DATA". Then you can do what you want with it - it is the data on the tone that came to you from the TON.