Ethereum RPC vs Peerplays RPC

Introduction

This document outlines differences between Ethereum and Peerplays RPC, with the goal to show changes we need to make in order to enable interoperability between public markets and Peerplays Non-fungible token implementation.

Non-fungible tokens (NFTs) are unique, digital items with blockchain-managed ownership. Examples include collectibles, game items, digital art, event tickets, domain names, and even ownership records for physical assets.

Our design will relly on EIP-721 standard, originating from Ethereum blockchain.

Original EIP can be found here:
https://github.com/ethereum/EIPs/blob/master/EIPS/eip-721.md

Ethereum RPC format

The example is taken from:
https://stackoverflow.com/questions/48228662/get-token-balance-with-ethereum-rpc

General format of CURL command for calling Ethereum function is this:

```
curl -X POST --data '{"jsonrpc":"2.0","method":"METHOD NAME","params":
[ENCODED FUNCTION SELECTOR AND PARAMS],"id":0}' -H "Content-Type:
application/json" NODE_ENDPOINT
```

For calling function that does not require transaction, eg balanceOf:

```
# We need to know contract address
# EOS token contract at
#    0x86fa049857e0209aa7d9e616f7eb3b3b78ecfdb0

# We are using eth_call
# params need to contain following fields
# - to: contract address
# - data: encoded function selector and function parameters
```

Function selector is calculated as first 4 bytes of sha3 hashed value of function signature - name and parameter type. E.g.

```
sha3("totalSupply(uint))
sha3("balanceOf(address))
sha3("allowance(address,address))

# We take first 4 bytes from these hashes, and thats our function
selector
```

Parameters are encoded as the big-endian two's complement encoding of X, padded on the higher-order (left) side with 0xff for negative X and with zero bytes for positive X such that the length is a multiple of 32 bytes. E.g.

```
integer value 10 is encoded as
    0x000000000000000000000000000000000000000000000000000000000000000A
boolean value true is encoded as
    0x0000000000000000000000000000000000000000000000000000000000000001
address value 0x0b88516a6d22bf8e0d3657effbd41577c5fd4cb7 is encoded as
    0x0000000000000000000000000b88516a6d22bf8e0d3657effbd41577c5fd4cb7
```

Data field for calling EOS token contract function balanceOf, for address 0x0b88516a6d22bf8e0d3657effbd41577c5fd4cb7 as parameter is encoded like this:

```
Function selector: 70a08231
Argument address:
0000000000000000000000000b88516a6d22bf8e0d3657effbd41577c5fd4cb7

data:
0x70a082310000000000000000000000000b88516a6d22bf8e0d3657effbd41577c5fd4c
b7
```

With all this a parameters for RPC call would be:

```
"params":["to": "0x86fa049857e0209aa7d9e616f7eb3b3b78ecfdb0", "data":
"0x70a082310000000000000000000000000b88516a6d22bf8e0d3657effbd41577c5fd4
cb7"]
```

So, a final curl command would look like this:

```
curl -X POST --data '{"jsonrpc":"2.0","method":"eth_call","params":
["to": "0x86fa049857e0209aa7d9e616f7eb3b3b78ecfdb0", "data":
"0x70a082310000000000000000000000000b88516a6d22bf8e0d3657effbd41577c5fd4
cb7"],"id":0}' -H "Content-Type: application/json" NODE_ENDPOINT
```

Peerplays RPC format

General format of CURL command for calling Peerplays function is this:

```
curl -X POST --data '{"jsonrpc": "2.0", "method": "METHOD NAME",
"params": [PARAMETERS], "id": 1}' -H "Content-Type: application/json"
NODE_ENDPOINT
```

For calling function that does not require transaction, eg get_account_balances:

```
# We need to know account ID and assets IDs whose balances we want to
retrieve.
# Main committee account ID is
#    1.2.0
# If we leave empty array for asset IDs, we will retrieve balances of
all assets owned by account
```

Parameters do not require any specific encoding, they are provided as a string in json notation (eg for arrays we use []). So, a final curl command would look like this:

```
curl -X POST --data '{"jsonrpc":"2.0","method":"get_account_balances","
params":["1.2.0",[]],"id":0}' -H "Content-Type: application/json"
NODE_ENDPOINT
```

Differences

The general format is pretty much the same, but parameter types and parameter count might differ. The most important difference is that there is no direct mapping between Ethereum and Peerplays parameter types and parameter count for any specific call. Peerplays does not have a notion of smart contract address. Peerplays uses object IDs to uniquely identify the objects, like users and assets, while Ethereum uses addresses for identifying users, and calls to contract addresses with encoded function and parameters are used for handling assets. Peerplays does not use function selectors for RPC calls, like Ethereum does, it uses function names. Peerplays does not use parameters encoding, like Ethereum does, it uses plain text.

Due to all these differences, it is very hard to implement Peerplays RPC calls that will be compatible with Ethereum RPC. Some kind of translator needs to be put in place in order to transform Ethereum RPC calls into Peerplays RPC calls, but due to big differences in data types, the final result is questionable.

Ethereum NFT vs Peerplays NFT