The Fluid Protocol

Progressive updating of blockchain parameters without hard forking

Executive Summary:

Duality Blockchain Solutions LLC is a blockchain as a service (BaaS) provider for our clients. As such, Duality needs a way to self-regulate the decentralized blockchain. The fluid protocol is the answer to the problem mentioned above without going through contentious hard forking scenarios. The core of the protocol, as of right now, can generate or mint new coins, change Dynode rewards, and change miner rewards. In the future, other additional changes to the consensus rules will follow.

Fluid Protocol Use Case

The market demand for alternative methods of money is present and here to stay with the advent of the Internet, the 2008 market crash, and overall lack of government accountability. Digital currencies are the answer to the rising needs of the market. In late 2008-2009, Bitcoin was the response of unaccountable government from the brilliant mind of Satoshi Nakamoto. While fast transactions, no middle-men, and cryptographic proofs provided transparency in Bitcoin, it did not offer the same level of features that the market wanted. The market enjoys regulatory oversight, uniform rules among a shared jurisdiction, and arbitration.

Duality Solutions is responding to such market needs with self-regulation, added arbitration features, and voluntary governance. Duality uniquely understands the shifting needs of the market investor, and the Fluid protocol is the solution to address those needs. The Fluid protocol is a mechanism to change the consensus rules of Duality's blockchain, Dynamic, to enforce self-regulation, change the reward amounts for Dynode holders and miners, and respond to arbitration.

From a technical level, there are five sovereign wallets, think of them as arbitrators, who need three out of five sovereign wallets signatures to make a decision. In essence, a counterparty mechanism to prevent abuse from

malicious users. In traditional banking and finance, banks and third parties are the ultimate approval mechanisms for transactions and changes within the financial network. The way to traverse and avoid the need for a final arbitrator is by spreading the power among multiple, vested parties. In digital currencies, this is called counterparty risk, and multi-signature wallets, which is an account where a specified amount of people out of the total amount are needed to move the funds out of the wallet, are the solution for addressing counterparty risk. The Fluid protocol is that for the Dynamic blockchain.

Technical Documentation

The writer assumes the reader has a basic understanding of the Dynamic blockchain, wallet addresses, and Dynodes.

The Fluid protocol gives “sovereign” wallet addresses control of the blockchain’s parameters, multi-signature addresses for the blockchain, using specific transactions. Usually, it requires a hard fork to change the consensus rules for a blockchain such as changing the rewards. With Fluid, static consensus rules change when 3 out of 5 sovereign wallet addresses sign a Fluid transaction. Fluid transactions perform the following actions in the current implementation:

- Generate or mint new coins
- Change Dynode rewards
- Change miner rewards

Future upgrades will allow Fluid to:

- Zero out a wallet address balance (by sending a negative txout).
- Ban a wallet address or Blockchain Directory Access Protocol (BDAP) account
- Adjust standard transaction and BDAP registration fees
- Add, remove, and swap sovereign address

Workflow: Currently, there are only RPC commands to create and view Fluid transactions.

To run the examples on testnet, you will need to import the private keys for at least three sovereign testnet wallet address. All Fluid transaction cost

100,000 DYN to run so you will need to verify you have the balance when you send the transaction in the last step.

Fluid RPC Calls

- burndynamic "amount" "account"
- consenttoken "address" "tokenkey"
- getfluidhistory
- getfluidhistoryraw
- getfluidsovereigns
- gettime
- getrawpubkey "address"
- maketoken "amount" "time" "receive address"
- sendfluidtransaction "hexstring"
- signtoken "address" "tokenkey"
- verifyquorum "tokenkey"

Fluid Protocol Code

All of the code for the fluid protocol can be found in these files:

- fluid.h
- fluid.cpp
- rpcfluid.cpp

Importing a Sovereign Wallet Address

The Dynamic testnet has 5 initial sovereign wallet addresses. This example shows how to import the sovereign private key:

- DSCex4e189aULrig3nLd42gVf7AbjTwnP5
- DMAh37n3RUdDxox3uiWAnc1zEPp5yFbHiL
- DN4KvqtXyygooPV3oha72TyBB5nqBbkxwj

```
# Import DSCex4e189aULrig3nLd42gVf7AbjTwnP5 sovereign wallet add
$./dynamic-cli importprivkey QVKXuZ2hSo2cT9BhkN3CApLuZYVsuzNvidJ
null
# Import DMAh37n3RUdDxox3uiWAnc1zEPp5yFbHiL sovereign wallet add
$./dynamic-cli importprivkey QU4VGDcVoej7nDZiyaSgoL7foG8xKiaVyk5
```

```
null
# Import DN4KvqtXyygooPV3oha72TyBB5nqBbkxwj sovereign wallet add
$./dynamic-cli importprivkey QWjTe6sCFVtKBsXfrYDyrHzn7eBeJktsQnW
null
```

Fluid Transaction Structure

`scriptPubKey = <OP_Code> <Instructions> <pubkey1 + sig> <pubkey2 + sig> <pubkey3 + sig>`

`OP_MINT <amount> <time> <receive address> <pubkey1 + sig> <pubkey2 + sig> <pubkey3 + sig>`

`OP_REWARD_DYNODE <amount> <time> <pubkey1 + sig> <pubkey2 + sig> <pubkey3 + sig>`

`OP_REWARD_MINING <amount> <time> <pubkey1 + sig> <pubkey2 + sig> <pubkey3 + sig>`

Where pubkey1, pubkey2, and pubkey3 are sovereign addresses.

Minting new coins on testnet

Example on how to mint 7,350 new DYN coins with fluid:

```
# get current Epoch time
$./dynamic-cli gettime
1531173072
```

```
# (Optional) Create a new wallet address.  You can use an existi
$./dynamic-cli getnewaddress
D5skCLLSk38sBNGKJYpaeTFkrboSzTTFdN
```

```
# Create the fluid mint token using amount minted, the epoch tim
# maketoken <amount> <time> <receive address>
$./dynamic-cli maketoken 7350 1531173072 D5skCLLSk38sBNGKJYpaeTF
373335302431353331313733303732244435736B434C4C536B333873424E474B
```

```
# Sign the token with one of five sovereign address
$./dynamic-cli signtoken DSCex4e189aULrig3nLd42gVf7AbjTwnP5 3733
```

```
373335302431353331313733303732244435736B434C4C536B333873424E474B
```

```
# Using the signed token above, consent the token with another s
$./dynamic-cli consenttoken DMAh37n3RUdDxox3uiWAnc1zEPp5yFbHiL 3
373335302431353331313733303732244435736B434C4C536B333873424E474B
```

```
# Using the signed token above, consent the token with a third s
$./dynamic-cli consenttoken DN4KvqtXyygooPV3oha72TyBB5nqBbkxwj 3
373335302431353331313733303732244435736B434C4C536B333873424E474B
```

```
# Check if the consent token above if valid and has all signatur
$./dynamic-cli verifyquorum 37333530243135333131373330373224443 5
Quorum is present!
```

```
# Using the consent token above, send the fluid transaction.  Fl
$./dynamic-cli sendfluidtransaction OP_MINT 37333530243135333131
A56994dd9654290cda929f9c6a9ba27cadabeaff6cec45ceb6baa09533123371
```

```
# Wait for fluid transactions to confirm in 2 blocks.
# Check fluid transactions
$./dynamic-cli getfluidhistory
[
  {
    "operation": "OP_MINT",
    "amount": "7350",
    "timestamp": 1531173072,
    "payment address": "D5skCLLSk38sBNGKJYpaeTFkrboSzTTFdN",
    "sovereign address 1": "DSCex4e189aULrig3nLd42gVf7AbjTwnP5",
    "sovereign address 2": "DMAh37n3RUdDxox3uiWAnc1zEPp5yFbHiL",
    "sovereign address 3": "DN4KvqtXyygooPV3oha72TyBB5nqBbkxwj"
  }
]
# View raw fluid transactions (hex results removed for readabili
```

```
# Example fluid mint coin transaction
$./dynamic-cli getrawtransaction a56994dd9654290cda929f9c6a9ba27
{
  "txid": "a56994dd9654290cda929f9c6a9ba27cadabeaff6cec45ceb6baa
  "size": 523,
```

```
  "version": 1,
  "locktime": 7983,
  "vin": [
    {
      "txid": "f5bf2e220b67cb0a17e290be694fff2402b5538368c515602
      "vout": 1,
      "scriptSig": {
        "asm": "3044022…."
    },
      "sequence": 4294967294
    }
  ],
  "vout": [
    {
      "value": 100000.00000000,
      "valueSat": 10000000000000,
      "n": 0,
      "scriptPubKey": {
        "asm": "OP_MINT 373335302431353331313733303732244435736b
        "type": "nonstandard"
      }
    },
    {
      "value": 10399775.12883341,
      "valueSat": 1039977512883341,
      "n": 1,
      "scriptPubKey": {
        "asm": "OP_DUP OP_HASH160 402327337a35255bbc8e49eb09ce2c
        "hex": "76a914402327337a35255bbc8e49eb09ce2c9153cbf58688
        "reqSigs": 1,
        "type": "pubkeyhash",
        "addresses": [
          "DAzDu1zDXXebz8GMj4c4ybjucVtis8U8iC"
        ]
      }
    }
  ]
}
```


Changing Dynode Rewards on testnet

Example on how to update the Dynode reward per block to 7.35 DYN coins with fluid:

```
# get current Epoch time
$./dynamic-cli gettime
1531173965
```

```
# Create the fluid mint token using amount minted, the epoch tim
# maketoken <amount> <time>
$./dynamic-cli maketoken 7.35 1531173965
372E333524313533313133393635
```

```
# Sign the token with one of five sovereign address
$./dynamic-cli signtoken DSCex4e189aULrig3nLd42gVf7AbjTwnP5 372E
372E333524313533313137333963540494E7645704F36634E454E58372B7547
```

```
# Consent the token with a second sovereign address
$./dynamic-cli consenttoken DMAh37n3RUdDxox3uiWAnc1zEPp5yFbHiL 3
```

```
# Consent the token with a second sovereign address
$./dynamic-cli consenttoken DN4KvqtXyygooPV3oha72TyBB5nqBbkxwj 3
```

```
# Check if the consent token above if valid and has all signatur
$./dynamic-cli verifyquorum 372E333524313533313137333963540494E
Quorum is present!
```

```
# Using the consent token above, send the fluid transaction.  Fl
$./dynamic-cli sendfluidtransaction OP_REWARD_DYNODE 372E3335243
59a3022a62ccc9b7e29ada7a042491250ae3dd7bfcb0c023708ec8122ef89ab7
```

```
# Wait for fluid transactions to confirm in 2 blocks.
# Check fluid transactions
$./dynamic-cli getfluidhistory
[
   {
    "operation": "OP_REWARD_DYNODE",
    "amount": "7.35",
    "timestamp": 1531173965,
    "sovereign address 1": "DSCex4e189aULrig3nLd42gVf7AbjTwnP5",
    "sovereign address 2": "DMAh37n3RUdDxox3uiWAnc1zEPp5yFbHiL",
    "sovereign address 3": "DN4KvqtXyygooPV3oha72TyBB5nqBbkxwj"
```

```
  }
]
```

```
# View raw fluid transactions
# Example change Dynode reward transaction
$./dynamic-cli getrawtransaction a56994dd9654290cda929f9c6a9ba27
{
  "txid": "59a3022a62ccc9b7e29ada7a042491250ae3dd7bfcb0c023708ec
  "size": 488,
  "version": 1,
  "locktime": 7992,
  "vin": [
    {
      "txid": "a56994dd9654290cda929f9c6a9ba27cadabeaff6cec45ceb
      "vout": 1,
      "scriptSig": {
        "asm": "3044022…."
      },
      "sequence": 4294967294
    }
  ],
  "vout": [
    {
      "value": 100000.00000000,
      "valueSat": 10000000000000,
      "n": 0,
      "scriptPubKey": {
        "asm": "OP_REWARD_DYNODE 372e333524313533313137333936354
        "type": "nonstandard"
      }
    },
    {
      "value": 10299775.12878451,
      "valueSat": 1029977512878451,
      "n": 1,
      "scriptPubKey": {
        "asm": "OP_DUP OP_HASH160 012b485de08215c077e63931e2cdfe
        "hex": "76a914012b485de08215c077e63931e2cdfee9529864eb88
        "reqSigs": 1,
        "type": "pubkeyhash",
        "addresses": [
          "D5FH4nQrJs4u3NEY3bCp1rjMMoftrXFb7C"
        ]
      }
    }
  ]
}
```

Changing Mining Rewards on testnet

Example on how to update the Dynamic mining reward per block to 1.9735 DYN coins with fluid:

```
# get current Epoch time
$./dynamic-cli gettime
1531174746
```

```
# Create the fluid mint token using amount minted, the epoch tim
# maketoken <amount> <time>
$./dynamic-cli maketoken 1.9735 1531174746
312E393733352431353331313734373436
```

```
# Sign the token with one of five sovereign address
$./dynamic-cli signtoken DSCex4e189aULrig3nLd42gVf7AbjTwnP5 312E
312E39373335243135333131373437343640494957 4C73675864484C53617978
```

```
# Consent the token with a second sovereign address
$./dynamic-cli consenttoken DMAh37n3RUdDxox3uiWAnc1zEPp5yFbHiL 3
```

```
# Consent the token with a second sovereign address
$./dynamic-cli consenttoken DN4KvqtXyygooPV3oha72TyBB5nqBbkxwj 3
```

```
# Check if the consent token above if valid and has all signatur
$./dynamic-cli verifyquorum 312E393733352431353331313734373436 40
Quorum is present!
```

```
# Using the consent token above, send the fluid transaction.  Fl
$./dynamic-cli sendfluidtransaction OP_REWARD_MINING 312E3937333
8969e02d255698577cc3671e3e80fc461ccddedf1a375541be6fe9140b1729ce
```

```
# Wait for fluid transactions to confirm in 2 blocks.
# Check fluid transactions
$./dynamic-cli getfluidhistory
[
```

```
  {
    "operation": "OP_REWARD_MINING",
    "amount": "1.9735",
    "timestamp": 1531174746,
    "sovereign address 1": "DSCex4e189aULrig3nLd42gVf7AbjTwnP5",
    "sovereign address 2": "DMAh37n3RUdDxox3uiWAnc1zEPp5yFbHiL",
    "sovereign address 3": "DN4KvqtXyygooPV3oha72TyBB5nqBbkxwj"
  }
]
```

```
# View raw fluid transactions
# Example change mining reward transaction
$./dynamic-cli getrawtransaction 8969e02d255698577cc3671e3e80fc4
{
  "txid": "8969e02d255698577cc3671e3e80fc461ccddedf1a375541be6fe
  "size": 490,
  "version": 1,
  "locktime": 8003,
  "vin": [
    {
      "txid": "59a3022a62ccc9b7e29ada7a042491250ae3dd7bfcb0c0237
      "vout": 1,
      "scriptSig": {
        "asm": "304402207…."
      },
      "sequence": 4294967294
    }
  ],
  "vout": [
    {
      "value": 100000.00000000,
      "valueSat": 10000000000000,
      "n": 0,
      "scriptPubKey": {
        "asm": "OP_REWARD_MINING 312e393733352431353331313734373
        "type": "nonstandard"
      }
    },
    {
      "value": 10199775.12873541,
      "valueSat": 1019977512873541,
      "n": 1,
      "scriptPubKey": {
        "asm": "OP_DUP OP_HASH160 c8befd2bfdb259a19df401d0e79673
        "hex": "76a914c8befd2bfdb259a19df401d0e7967382e9bd732188
        "reqSigs": 1,
        "type": "pubkeyhash",
```

```
        "addresses": [
          "DPSYUmb8o9oPJr9j8p6HVjfMkMnaJpGCUg"
        ]
      }
    }
  ]
}
```