NavCoin Community Fund Technical Paper

alex v

July 4, 2018

1 Introduction

The Community Fund has been successfully approved by the NavCoin network. Before rolling it out definitively on the network, a previous beta phase must be passed to ensure the safety and correctness of the changes introduced in the consensus protocol.

2 Protocol

This part describes solely the technical insights on how the Community Fund is implemented in the NavCoin network. Implementation of RPC commands, updating the wallet UI to facilitate the different functionalities and creating a bulletin board to host the discussion around Proposals are later tasks which require a different discussion and study, being some of them commented in next parts of this document.

2.1 New OP Codes

```
OP_CFUND = 0xc1
OP_PROP = 0xc2
OP_PREQ = 0xc3
OP_YES = 0xc4
OP_NO = 0xc5
```

2.2 Activation of the Community Fund

Readiness of nodes to process the new consensus rules would be signalled setting the bit n.6 of the block version. When a majority (more than 75%) of blocks signal for readiness in a defined period of blocks (20160 blocks = 1 week) the community fund will be activated and new rules will apply:

- Every block must add an extra output at the end of the coinstake tx (second transaction of a block) and should keep signalling readiness even after activation point.
- Blocks which do not signal readiness for the community fund will be rejected by the network.
- ScriptPubKey of the tx must be OP_RETURN OP_CFUND
- nValue of the tx must be 0.25 NAV (25000000 navoshis).
- Yearly staking rewards must be reduced to 4%.

2.3 Definition of Actors

Staker: a participant in the network who holds coins in its wallet and takes part in the staking process to secure and mint new blocks. Those can be divided between:

- Active Stakers: Stakers which get involved in the voting process and decide to vote on Proposals.
- Passive Stakers: Those who decide to only participate securing the network, collecting staking rewards and do not get part on the Proposal votings.

Any staker can freely morph from active to passive state and vice versa by the solely act of voting or non voting.

Only votes from active stakers will be weighted into the results of the votings.

Proponent: any user, staker or external actor who presents a Proposal to withdraw coins from the community fund.

2.4 Creation of Proposals

A Proposal will be identified by the hash of a transaction. This transaction must:

- Have an output with scriptPubKey == OP_RETURN OP_CFUND and a variable amount taken as a fee to prevent spamming. Consensus rules will determine a minimum fee. The amount will be added to the community fund.
- Have nVersion == 4 as defined in constant Proposal_VERSION (src/primitives/transaction.h)
- Include in the strDZeel field of the transaction basic metadata about the Proposal:

```
{ "v" : version of the Proposal,
  "n" : required amount,
  "a" : address for the payouts,
  "d" : deadline for the Proposal in epoch format,
  "s" : string with a min. description of the Proposal }
```

2.5 Creation of Payment Requests

A Payment Request will be identified by the hash of a transaction. This transaction must:

- Have an output paying a minimum network fee of 10000 Navoshis.
- Have nVersion == 5 as defined in constant PAYMENT_REQUEST_VERSION (src/primitives/transaction.h)
- Include in the strDZeel field of the transaction basic metadata about the parent Proposal and the Payment Request:

```
{ "v" : version of the Payment Request,
  "h" : hash of the parent Proposal,
  "n" : required amount,
  "s" : proof of ownership of the parent Proposal deposit
        address,
  "i" : unique id used to identify the Proposal }
```

"s" must be the result of signing

```
I kindly ask to withdraw required_amount NAV
from the proposal parent_proposal_hash.
Payment request id: proposal_identifier_i
```

using the Proposal's deposit address.

2.6 Voting of Proposals

A staker will add commitments in the coinbase (first transaction of a block) of its minted blocks in the form of extra outputs for every entry in his Proposals hash list. Those outputs will meet following structure:

- scriptPubKey (36 bytes): OP_RETURN OP_CFUND OP_PROP OP_YES ProposalHash
- nValue: 0

Stakers will be able to vote against a Proposal including its hash in a list of rejected Proposals and modifying the structure of the appropriate output of the coinbase commitment:

- scriptPubKey (36 bytes): OP_RETURN OP_CFUND OP_PROP OP_NO ProposalHash
- nValue: 0

2.7 Voting of Payment Requests

Stakers will be able to express his vote for releasing funds from any withdrawal request adding the following commitment as an output to the coinbase transaction:

Positive vote:

- scriptPubKey (36 bytes): OP_RETURN OP_CFUND OP_PREQ OP_YES PaymentRequestHash
- Amount: 0

Negative vote:

- scriptPubKey (36 bytes): OP_RETURN OP_CFUND OP_PREQ OP_NO PaymentRequestHash

- Amount: 0

2.8 Release of funds

Stakers will add payments of approved Payment Requests as extra outputs of the coinbase transaction. Those will meet following structure:

- scriptPubKey: Normal P2PKH script for the NavCoin Address of the parent Proposal of the Payment Request.
- nValue: The value of the approved Payment Request.

In top of this, stakers will include in the strDZeel field of the coinbase transaction an array in format JSON including the hashes of the Payment Requests the outputs are refering to, in strict order.

```
[ "Payment Request Hash 0",
  "Payment Request Hash 1",
  "Payment Request Hash 2" ]
```

Following the consensus rules, nodes will check for every output with nValue greater than 0 that the Payment Request referenced in the corresponding index of the array result of decoding the strDZeel, is conveniently approved and has not already been paid out.

3 Voting rules

Currently the Community Fund is using temporary values which allow to test it minimizing fees and reducing waiting time. The definitive values must reach consensus in the community before deploying the Fund on main net. The NavCoin Core team proposes the following values:

- Voting cycle (Vc): 2880 * 7 blocks (Approx 1 week)
- Min Quorum per period: 50% of a voting cycle
- Proposals/Payment Requests min age: 5 blocks
- % of Positive votes to accept a Proposal: 75%
- % of Negative votes to reject a Proposal: 67.5%
- % of Positive votes to accept a Payment Request: 50%
- % of Negative votes to reject a Payment Request: 50%
- Fee to create a Proposal: 100 NAV
- Maximum number of full elapsed Voting Cycles for a Proposal: 4 Voting Cycles (1 Month)
- Maximum number of full elapsed Voting Cycles for a Payment Request: 6 Voting Cycles (1 Month and a half)

Every block, both positive and negative count of the votes attached to the Proposals and Payment Requests will be updated summing any new vote included in the block. Every Vc blocks, the state of the Proposal/Payment Request will be updated based on the stored count of votes.

3.1 Proposals

A Proposal can have 5 different states:

3.1.1 Pending

Default state of a Proposal. Votes will only be casted when on this state.

3.1.2 Accepted

- When a voting cycle reaches its end and the proportion of positive votes over the sum of positive and negative votes is greater than 75%, and the sum of positive and negative votes is greater than the half of the length of the voting cycle, a Proposal is flagged as accepted.

A Proposal in this state can only mutate to state 'Expired'.

3.1.3 Rejected

- When a voting cycle reaches its end and the proportion of negative votes over the sum of positive and negative votes is greater than 67,5%, and the sum of positive and negative votes is greater than the half of the length of the voting cycle, a Proposal is flagged as rejected.

This state can not mutate.

3.1.4 Expired

- For Proposals with version 1: When a voting cycle reaches its end and the timestamp of the last confirmed block is higher than the Proposal's deadline, the Proposal is flagged as expired.
- For Proposals with version greater than 1: When 4 full voting cycles are elapsed without reaching consensus on whether the Proposal is accepted or rejected or when a voting cycle reaches its end and the timestamp of the last confirmed block is higher than the timestamp of the block when it was accepted summed to the Proposal's deadline, it will be flagged as expired.

This state can not mutate.

3.1.5 Pending Funds

- When a voting cycle reaches its end and the proportion of positive votes over the sum of positive and negative votes is greater than 70%, and the sum of positive and negative votes is greater than 90, but the amount of coins available on the fund is less than the amount requested by the Proposal, the Proposal will be flagged as Pending Funds.

This state can change to 'Accepted' if enough coins are available at the end of next voting cycle or to 'Expired' if the conditions regarding the timestamp required for a Proposal to be expired are met.

3.2 Payment Requests

Owners of accepted Proposals can request payments 5 blocks after the end of the voting cycle where it was accepted as defined in section 3.5. A Payment Request can only receive votes while its parent Proposal is in accepted state. For Payment Requests with version greater than 1, it'll be enough for the Proposal to be in accepted state at the moment of the Payment Request being created. There are three possible states, none of them can mutate:

3.2.1 Accepted

When a voting cycle reaches its end and the proportion of positive votes over the sum of positive and negative votes is greater than 50%, and the sum of positive and negative votes is greater than the half of the length of the voting cycle, a Payment Request is flagged as accepted. 5 blocks after the end of the voting cycle a payment of the requested amount will be released to the address specified in the parent Proposal as defined in section 3.8.

3.2.2 Rejected

When a voting cycle reaches its end and the proportion of negative votes over the sum of positive and negative votes is greater than 50%, and the sum of positive and negative votes is greater than the half of the length of the voting cycle, a Payment Request is flagged as rejected.

3.2.3 Expired

When 4 full voting cycles are elapsed without reaching consensus on whether the Payment Request is accepted or rejected, it is flagged as expired. This state exists only for Payment Requests with versions greater than 1.

4 New RPC commands

During the beta phase the only way to interface with the Community Fund is using RPC commands. Next sections will describe the new commands introduced.

4.1 createProposal

Syntax: createProposal address amount deadline

Creates a Proposal for the community fund. Min fee of 0.000100NAV is required.

Arguments:

- "navcoinaddress" (string, required) The navcoin address where coins would be sent if Proposal is approved.
- "amount" (numeric or string, required) The amount in NAV to requesst. eg 0.1
- deadline (numeric, required) Number of seconds the Proposal will exist after being accepted.
- "desc" (string, required) Short description of the Proposal.
- fee (numeric, optional) Contribution to the fund used as fee.

Result:

```
{ hash: Proposalid,           (string) The Proposal id.
strDZeel: string }            (string) The attached strdzeel property.
```

Examples:

```
navcoin-cli createProposal "NQFqqMUD55ZV3PJEJZtaKCsQmjLT6JkjvJ"
1000 1509151016 "Development"
```

4.2 createpaymentrequest

Syntax: createpaymentrequest hash amount id

Creates a Proposal to withdraw funds from the community fund. Fee: 0.0001 NAV

Arguments:

- "hash" (string, required) The hash of the Proposal from which you want to withdraw funds. It must be approved.
- "amount" (numeric or string, required) The amount in NAV to withdraw. eg 10
- "id" (string, required) Unique id to identify the Payment Request

Result:

```
{ hash: prequestid,           (string) The Payment Request id.
strDZeel: string }            (string) The attached strdzeel
property.
```

Examples:

```
navcoin-cli createpaymentrequest "196a4c2115d3c1c1dce1156eb2404ad7
7f3c5e9f668882c60cb98d638313dbd3" 1000 "Invoice March 2017"
```

4.3 getProposal

Syntax: getProposal "hash"

Shows information about the given Proposal.

Arguments:

- hash (string, required) the hash of the Proposal

4.4 getpaymentrequest

Syntax: getpaymentrequest "hash"

Shows information about the given Payment Request.

Arguments:

- hash (string, required) the hash of the Payment Request

4.5 listProposals

Syntax: listProposals "filter"

Shows the list of Proposals. Filter can be:

- accepted
- rejected
- expired
- pending

4.6 Proposalvote

Syntax: Proposalvote Proposal_hash command

Add or remove a Proposal to the list of votes.

Arguments:

- "Proposal_hash" (string, required) The Proposal hash
- "command" (string, required) 'yes' to vote yes, 'no' to vote no, 'remove' to remove a Proposal from the list

4.7 Proposalvotelist

Syntax: Proposalvotelist Shows a list of your votes for Proposals.

4.8 paymentrequestvote

Syntax: paymentrequestvote payment_request_hash command

Add or remove a Payment Request to the list of votes.

Arguments:

- "payment_request_hash" (string, required) The Payment Request hash
- "command" (string, required) 'yes' to vote yes, 'no' to vote no, 'remove' to remove a Proposal from the list

4.9 paymentrequestvotelist

Syntax: paymentrequestvotelist Shows a list of your votes for Payment Requests.

4.10 cfundstats

Syntax: cfundstats Returns information about the community fund and the current voting cycle.

Example:


```
{
  "funds": {
    "available": 469.50000000,
    "locked": 0.00000000
  },
  "votingPeriod": {
    "starting": 2520,
    "ending": 2700,
    "votedProposals": [
      {
        "str": "test",
        "hash": "693214130f7c13df102b2e7fcc0b14842514
6f01f19d0a969218d9b4b90a19d1",
        "amount": 100000000000,
        "yes": 102,
        "no": 0
      }
    ],
    "votedPaymentrequests": [
    ]
  }
}
```


4.11 donatefund

Syntax: donatefund amount (subtractfeefromamount)

Donates an amount to the community fund.

Arguments:

- "amount" (numeric or string, required) The amount in NAV to donate. eg 0.1

- subtractfeefromamount (boolean, optional, default=false) The fee will be deducted from the amount being sent. The fund will receive less navcoins than you enter in the amount field.

Result: "transactionid" (string) The transaction id.

Examples:

```
navcoin-cli donatefund 0.1
navcoin-cli donatefund 0.1 true
```

5 Reporting bugs and communication

GitHub issues `https://github.com/NAVCoin/navcoin-core/issues` will be used to report bugs on the protocol. A new Discord channel '#communityfund-beta' has been created to coordinate the beta phase.