[This is a followup to @djrtwo's

[Simple eth1 withdrawals (beacon-chain centric)

](https://ethresear.ch/t/simple-eth1-withdrawals-beacon-chain-centric/8256) and provides several ideas on top]

Also thanks @mkalinin for review

What we already have:

- deployed Deposit contract, we cannot rollback this
- contract verifies that withdrawal_credentials

are of 32 bytes length, so we couldn't put anything longer in it using current deposit contract

- only withdrawal_credentials

started with 0x00

are committed (with hash of BLS withdrawal public key), though contract doesn't verify it, so we could use other prefixes without changes in contract

What do we want:

Make withdrawals usable both for individual users and different kinds of pools, both custodial and non-custodial. In best case pool assets should be tokenized to fulfill investor's needs, so it would be easy to invest fractional values less than 32 ETH and exit to money whenever requested by each co-investor. We expect validator stake could be used for issuing derivative assets secured by stake in DeFi or similar, attract more funds in staking and increase development of derivative financial instruments in Ethereum. Pools and tokens should be secured by a well proven mechanism of Ethereum contracts to meet the needs of both investors and pool owners without much trust to each other.

How could we do it:

Push vs Pull

PUSH:

- Limited receive contract capabilities
- only default method could be called
- what gas limit should be set?
- only default method could be called
- what gas limit should be set?
- User is limited in handling of withdrawal costs
- Awkward errors handling (if failed, how to pull?)

PULL:

- Someone should send TX or call System Contract in Eth1 to initiate
- Caller could manage withdrawal result and execute any logic on top

With the target of keeping core Beacon Chain logic as simple as possible, PUSH with the same capabilities will be more complex, therefore PULL proposed for 0x01

-prefix should be preferred.

Pull, 0x01

prefix

This idea is explained by @djrtwo in Simple eth1 withdrawals (beacon-chain centric). We have following flow on withdrawals from user's side:

What is important, it also works for trustless pools:

[

798×503 22.9 KB

](https://ethresear.ch/uploads/default/original/2X/3/33f7a0ea7f76221bdebf3308dc687b83460735c2.png)

BLSSetWithdrawalAddress

New event is proposed for Eth2, BLSSetWithdrawalAddress

which could change withdrawal_credentials

. Some details about its handling:

- we verify that current validator's withdrawal_credentials

is started with BLSWithdrawalPrefix

, 0x00

- we verify that new credentials are started from supported prefixes and are not BLSWithdrawalPrefix

, 0x00

What this means for usage:

- Validator's BLS withdrawal key couldn't be changed
- 0x00

prefix in withdrawal_credentials

doesn't mean any particular withdrawal target, it means that withdrawal target will be submitted later and should be signed by appropriate key

- Exact withdrawal_credentials

(not 0x00

) could be set only once by submitting BLSSetWithdrawalAddress

at any time even after withdrawable_epoch

. We check whether validator is eligible for withdrawal right now in the end of processing

- Once additional prefixes are confirmed, they could be used in withdrawal_credentials

while making deposit by Deposit Contract call

- Once withdrawable_epoch

is hit by validator, we check whether withdrawal_credentials

are set to smth starting with a prefix other than 0x00

. If it's, process it, otherwise doing nothing

Some opposes and defends on this decisions:

N:

What if BLS withdrawal key is compromised, I should be able to change it

A:

You could set an exact withdrawal target in this case, since you are protected from anything signed by a compromised key.

N:

What if I set withdrawal to contract address in eth1 and its compromised

A:

You could submit a withdrawal contract address change signed by BLS key at any time, so the safest way is to do it, when you are confident in the withdrawal target. If it's allowed to change withdrawal credentials from one eth1 address to another, the dilemma appears, who should sign such a change. Both pools and their users could could trick each other if it's allowed.

NN:

We could add is_final

flag, so when it's false, you could submit another change

AA:

If it's not final you could submit a change signed by withdrawal key and change it, so not a final field secures nothing. So it's the same as having withdrawal_credentials

started with BLSWithdrawalPrefix

but without extra complexity.

Partial withdrawals

There are several concerns about partial withdrawals, let's dive into it. The main challenge of developing withdrawals is non-symmetry of eth2 with eth1. While it looks like both networks have some kind of accounts and balances associated with it, there are a lot of nuances

- BeaconState

and Ethereum 2 don't have any kind of transaction-like mechanics, so if it's introduced, we should care both about security and speed. We have about 15 TPS in Ethereum 1, and we target hundreds of thousands validators in Ethereum 2. Partial withdrawals, if available, could increase BeaconChain

load or even delay more important messages.

- Ethereum 2 events are not subject to any fees. It guarantees fastest and equal processing and is part of network safety requirements. Increasing the number of events dramatically could affect other Ethereum 2 events processing and so whole network safety.

- BeaconState

is not going to be stateless. It's the minimal state your client should have to be in Ethereum 2. And we expect that clients could be mobile clients, IoT clients etc in the future. So the target is to keep BeaconState

as small as possible, with the smallest number of changes on each slot/epoch.

What's on the other side. Do we really need it?

- Validator could exit and withdraw whenever it wants in less than 24 hours, it's fairly good liquidity for financial instrument

- Reward payment schedule is similar to some kinds of financial instruments like bonds and bank deposits, though both could provide intermediate payments before the end date.

- If withdrawal is built to make possible construction of pools with tokenization of investment, end user gets a contract which is traded and its price grows since the first date, so rewards could be collected before end date minus some premium

- There is minor interest drop due to the lack of partial withdrawals, but it's the same for all validators

Interest drop estimations

Let's estimate interest rate drop by sticking with only full withdrawal:

- Resign rate drop: (55 hours delay is used) If resigns happen less frequently than every 7 months drop is under 1% of advertised rate (for example, 3.96% instead of 4% or 19,80% instead of 20% on start).

- Compound interest rate drop: If we take same 7 months, 0.05% annualized rate miss for 4% annual reward rate (4.05% finally), 1.33% annualized rate miss for 20% of annual reward rate (21.33% finally), so it dramatically increases with speculation reward rates, but in mature network it's less dominant. At the moment reward is already under 15%.

- Inability to invest funds rate drop: It's difficult to give a range, as in the worst case, when you have 31.99 ETH and that's all, you lose all rewards, and with 32 ETH you get all possible rewards. The main rule here is: the less ETH you have the more you lose, but even when you have 351.99 ETH, you have almost a 10% drop rate (3.63% instead of

4%) because 31.99 ETH is not stacked.

So our main target should be making pools and tokenized stacking available as it will abolish the latter case. It will guarantee the decentralization of end users, co-investors of validators, small stake holders. Opportunity of making possible tokenization is more important than partial withdrawal.

Optional infrastructure

Eth1 -> Eth2 gate would be helpful (so, you could send BLSSetWithdrawalAddress

and anything else by call in Eth1) for pools and other contracts, though we are not sure there is a sense of making a system contract for this. Instead, any third party could create one and provide a service for a fee. Being off chain it doesn't require too much complexity and will offer services that system contracts are unable to provide, for example, callback. In the worst case you will lose a fee, which is small and good service will compensate for the outage by something which couldn't be written in Solidity.

While such service could improve simplicity of building Eth2 pools and staking tokenization contracts, it's not a requirement.

Executable beacon chain

One of the latest views on merging Ethereum 1 into Ethereum 2,A rollup-centric ethereum roadmap by @vbuterin assumes one Ethereum 1 execution shard. As Withdrawal touches both Ethereum 2 state and Ethereum 1 shard we expect asynchronous execution of withdrawal (we need to connect eth1 and eth2 events, handle errors, etc) in this case with a lot of extra complexity added due to this. If we are going to couple Ethereum 1 shard with Ethereum 2 state strongly, as currently proposed(see Vitalik's comment in AMA and Executable beacon chain), so Ethereum1 is wrapped by Ethereum2, it greatly reduces implementation complexity. While it could make eth1-eth2 communication way simpler and increases profitability of validators, it has several nuances:

- Eth1-Eth2 message flow becomes way simpler, so and 0x01

withdrawal prefix implementation

- Strong coupling of Beacon Chain and one Ehereum execution shard opens space for almost symmetric deposit-withdrawal designs
- If tight coupling of Ethereum1 shard is chosen for network architecture there is more incentive to build Eth1-Eth2 gate mentioned above as a System Contract, so you could be able to create and push Eth2 message from Ethereum 1 contract code without 3rd party trust
- Eth1 TX processing and fee collection will be a part of validator's job, so staking pools should aggregate both this fees and Eth2 validator withdrawals, which could be challenging, but total profit increases

Next steps

As @djrtwo stated, we need to only commit new prefix(es) today, and could start polishing the withdrawal specification. On the other side, this will be enough to start implementation contracts and pools. Though as start is already and withdrawal takes less than a day, we will definitely have a huge backlog of ready for withdrawal validators at launch of withdrawal feature, which should be handled. BLSSetWithdrawalAddress

could be added before everything else, as nothing prevents to submit withdrawal_credentials

to Deposit Contract today with a prefix different to 0x00

.

To prove viability of 0x01

withdrawal prefix we are going to make PoC of staking pool with fractional shares of validator's stake to clarify challenges in eth1-eth2 deposit-withdrawal flow from staking pool view and start designing of system contract.