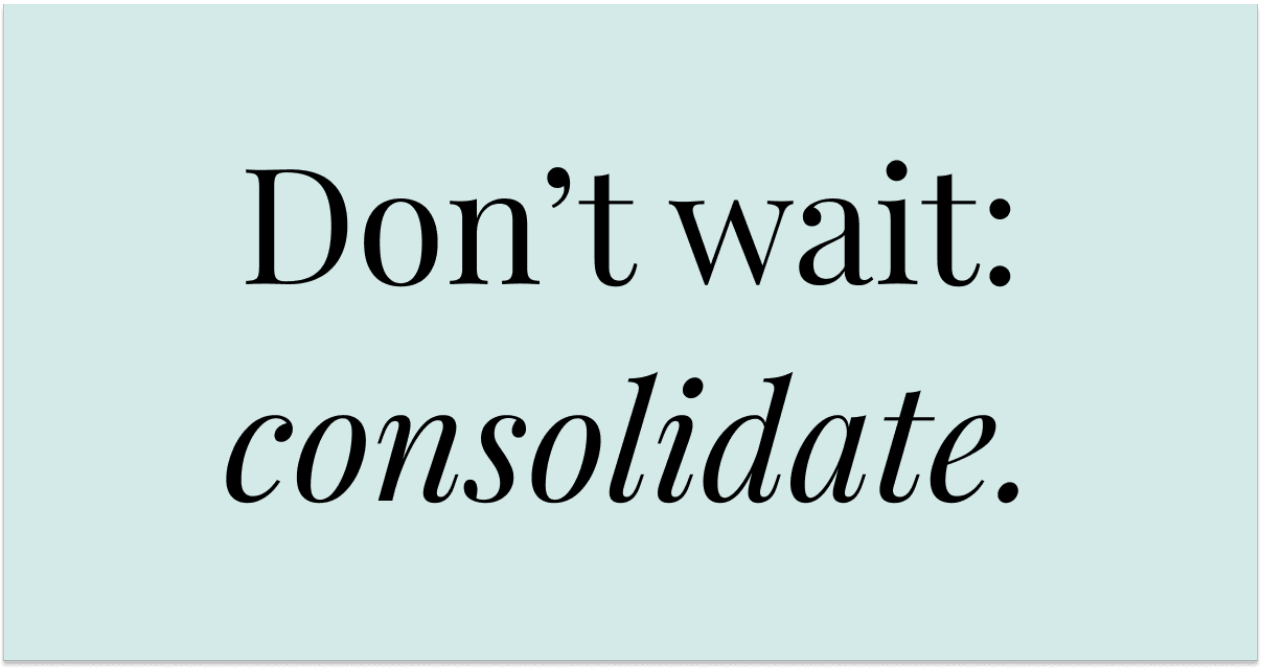
Validator consolidation in EIP-7251



$\cdot$

by francesco, mike, & mikhail

-- based on extensive discussions with Lion, Roberto, Sandra, & Barnabé

-- tuesday, january 23, 2024

$\cdot$

***tl;dr;** EIP-7251 increases the MAX_EFFECTIVE_BALANCE of Ethereum validators to 2048 ETH while preserving the 32 ETH minimum. "In-protocol consolidation", a key feature in the EIP, allows validators to merge their effective balances (thus becoming a single validator with the combined stake) \~without\~ fully exiting and re-entering the protocol through the activation queue. This feature is critical to enable large staking pools that wish to merge validators because it allows them to do so without sacrificing significant rewards during the consolidation period. Due to the importance of this feature, we present an FAQ document that aims to answer the natural questions about the process and justify the design decisions.*

$\cdot$

EIP-7251: related work

$\cdot$

What and why of in-protocol consolidation

What is in-protocol consolidation?

- EIP-7251 increases the MAX_EFFECTIVE_BALANCE for validators in the Ethereum consensus layer. This does not affect the 32 ETH minimum needed to join the validator set, but rather it raises the maximum balance validators can have to 2048 ETH. See the related work for more context. To facilitate the adoption of the EIP, we are proposing a mechanism by which validators can merge without fully exiting and re-entering the consensus layer. The consolidation process allows a source validator to designate a target validator as the recipient of its effective balance. The consolidation is authenticated through signatures from both the source and target validators and asserts that the merging is authorized by checking that both validators have the same withdrawal credential.

Why do we need in-protocol consolidation?

- EIP-7251 is most effective if the large staking pools opt in to consolidating their stake onto fewer validators. Without an in-protocol consolidation mechanism, they would be required to fully exit their validators and redeploy the stake in a single larger validator (e.g., voluntarily exit 10 different 32 ETH validators to activate a single 320 ETH validator). This results in the 320 ETH sitting in the activation queue and not earning rewards for the pool, strongly disincentivizing consolidation – this is particularly painful if the activation queue becomes full again. If instead, they can signal their intent to merge validators through the protocol, that stake can skip the activation queue and continue earning rewards, despite being reallocated to a different validator.

What is the endgame for in-protocol consolidation?

- In-protocol consolidation is an interim solution to a problem (many redundant validator indices) that only exists because of the historical context of the original value of the MAX_EFFECTIVE_BALANCE being set at 32 ETH. Once node operators and stakers have the flexibility to deploy validators with variable sizes according to their infrastructure configurations and personal preferences, the consolidation function will likely no longer be critical. For example, instead of deploying two 32 ETH validators and consolidating them through the protocol, we would expect the staker to deploy a single 64 ETH validator (or start with a 32 ETH validator and add a deposit of another 32 ETH later). Despite this reality, in the short to medium term, this feature is by far the most important in terms of getting consolidation to take place.

Why should we do EIP-7251 if we can't guarantee consolidation?

- As mentioned above, consolidation will be opt-in. This begs the question of the benefits of EIP-7251, given the worst-case scenario is no consolidation takes place (rendering the EIP essentially a no-op). In our opinion, it is still worth doing EIP-7251 for three key reasons.

    1. Reason 1: we can expect some amount of initial consolidation. First and foremost, node operators can benefit from consolidation because it reduces the number of validators that they need to manage. Coinbase, for example, runs at least 139,000 validators. This is a lot of key management and reducing this complexity could reduce their OpEx significantly. There are currently $\approx 900,000$ validators in the set; in a fully-consolidated world, this would be reduced to $64x$ to $\approx 14,000$ validators. In reality, we expect a much more modest amount of consolidation, but even $900,000 \rightarrow 600,000$ would be a huge accomplishment and would set us up well to handle a large influx of new capital in the case that staking demand increases significantly during the next market cycle.

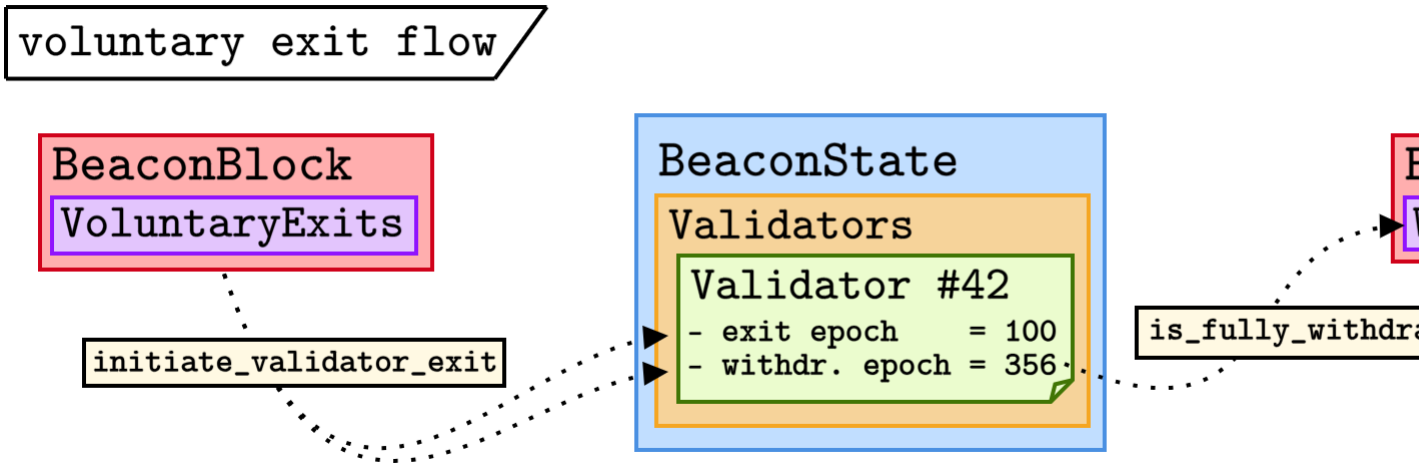
    2. Reason 2: Solo staking is significantly improved. Reducing the overall validator set size through large validator consolidation is the big ticket item for EIP-7251, but the solo staker UX improvements should not be overlooked. First, solo stakers immediately can start to benefit from auto-compounding of their staking rewards. Rather than needing to deploy another 32 ETH validator, the solo stakers may immediately begin earning rewards on any balance they accumulate above the 32 ETH principal (this is accomplished by the solo staker updating their withdrawal credential to indicate that they would like to avoid their excess balance being swept off, thus opting into compounding rewards). Historically, auto-compounding has only been accessible through joining a staking pool. Additionally, flexible balances allow solo stakers who have some amount of capital beyond the 32 ETH that they would like to stake (e.g., someone with 40 ETH can solo stake it all rather than needing solo staking 32 ETH and delegating 8 ETH to a staking pool).

    3. Reason 3: Variable balances are always going to be part of the Ethereum endgame. The endgame of Ethereum depends on single-slot finality. As Vitalik recently wrote, depending on continued optimization of BLS signature aggregation doesn't seem like a viable path forward. Variable validator balances are a core piece of a Proof-of-Stake protocol, so enshrining it now is future-compatible and forward-looking. If we later choose to incentivize further consolidation (either through a carrot or a stick), that discussion can be detangled from the exact mechanics of increasing the MAX_EFFECTIVE_BALANCE and stand alone as a much simpler EIP.

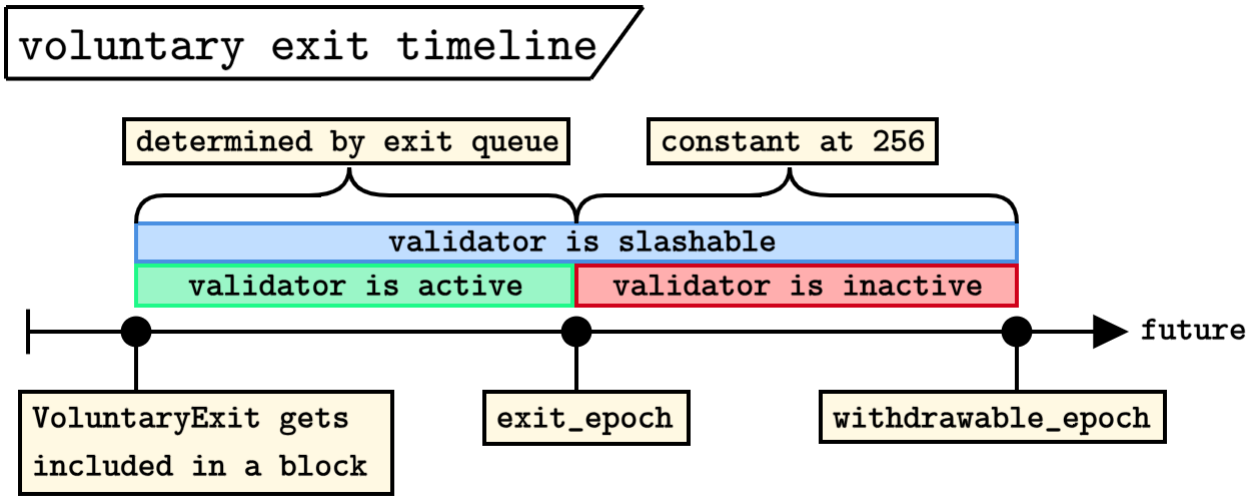
How in-protocol consolidation works

How does a voluntary exit work again? (refresher)

- Voluntary exits are not changed with EIP-7251, but it is helpful to have the process in mind when explaining the consolidation details. The figure below demonstrates the flow of a voluntary exit.

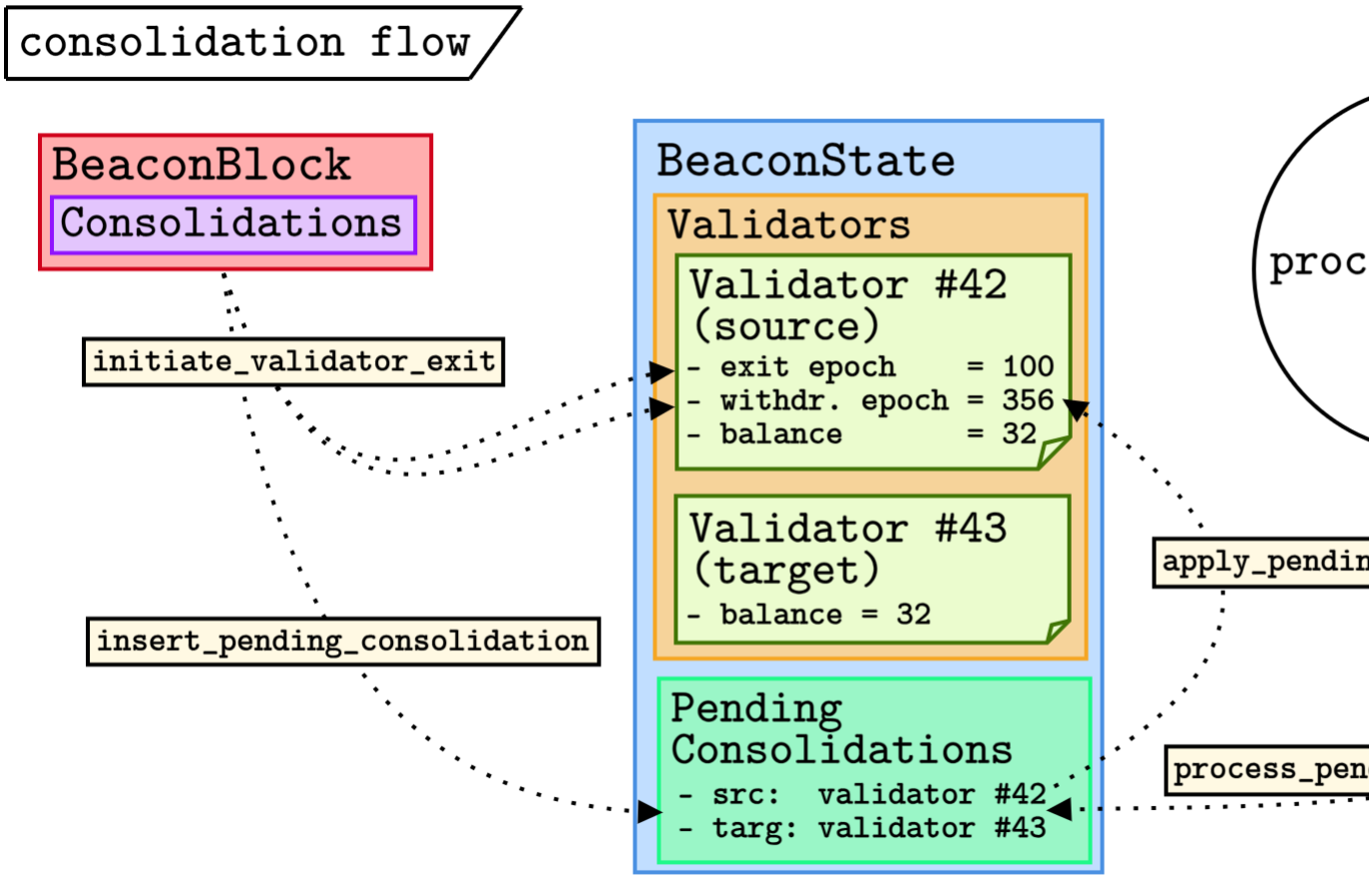


- Once the VoluntaryExit is included in the BeaconBlock, the initiate_validator_exit function is called, which sets the exit_epoch and withdrawable_epoch for the validator in the BeaconState. The exit_epoch depends on the churn limit, and the validator must continue performing their consensus duties until this epoch. The withdrawable_epoch is just the exit_epoch + MIN_VALIDATOR_WITHDRAWABILITY_DELAY, where MIN_VALIDATOR_WITHDRAWABILITY_DELAY=256 is a spec-defined constant. The validator is slashable until the withdrawable_epoch is reached, at which point their withdrawal is processed and their ETH is sent to the address specified in their withdrawal credential. The figure below shows the timeline of events for a voluntary exit.



What is the in-protocol consolidation process?

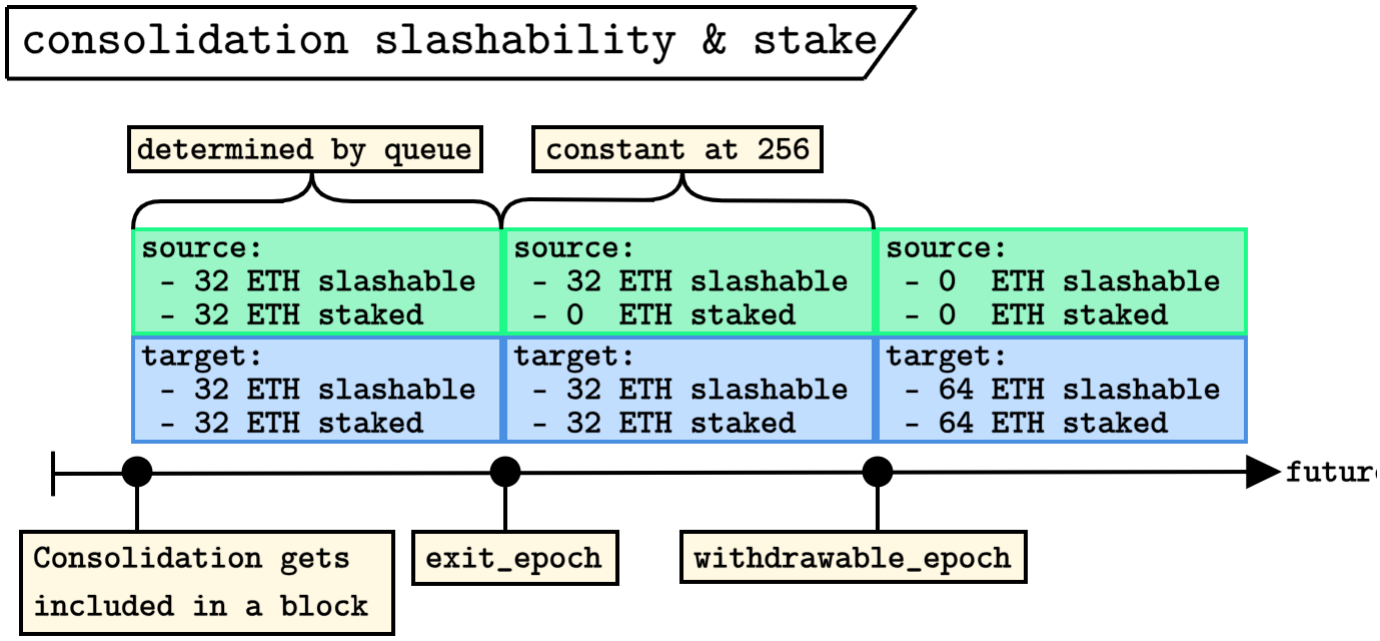
- In the same way that a validator broadcasts their VoluntaryExit message and it gets included in a block, both validators that would like to consolidate sign a Consolidation object and gossip it over the p2p layer. Once this consolidation is included in a block, it is processed and the merging of the two validators begins. An exit for the source validator is triggered and once the exit is complete, their balance is fully allocated to the target validator, bypassing the activation queue. *Crucially, as with normal exits, the source validator is active and keeps earning rewards while in the exit queue* The figure below demonstrates this flow.



- When the consolidation in the block is processed, the source validator's voluntary exit is triggered. Additionally, a pending consolidation is inserted into the `BeaconState`. When the epoch processing takes place, any pending consolidation where the `withdrawable_epoch` has been reached transfers the balance to the `target` validator. If by the time `process_pending_consolidation` is called in the `withdrawable_epoch`, the `source` validator has been slashed, then the consolidation operation is canceled.

What are the implications for the `source` and `target` validator rewards during a consolidation period?

- Let's define two phases of the consolidation process.
  1. *`source` exit phase*. The epochs between the `Consolidation` object being included onchain and the assigned `exit_epoch` for the `source` validator. The length of this phase depends on the exit queue.
  2. *`source` withdrawal phase*. The 256 epochs (as defined by `MIN_VALIDATOR_WITHDRAWABILITY_DELAY`) between the `exit_epoch` and `withdrawable_epoch` of the `source` validator.
- During (1), the `source` validator is earning rewards and participating in consensus. During (2) the `source` validator is not earning rewards nor participating in consensus. The `target` validator is earning rewards during both (1) and (2). *Note that this implies the consolidating validator is not earning rewards during the 256-epoch period between the exit and the withdrawal. This is a fixed amount of time, about 27 hours, independent of the size of the exit queue.* The figure below shows the slashability and stake of two consolidating validators.



How is slashing attributed during a consolidation?

- During the consolidation period, each validator is slashable for the stake that they each had initially (32 `ETH` in the base case). Once the exit period has ended, the `source` validator can safely stop participating in consensus (they are no longer active validators), but they are still slashable until the `withdrawable_epoch` (as with a normal voluntary exit). Once the `withdrawable_epoch` is reached, their balance is instantaneously transferred to the `target` validator, who is now accountable for the full combined balance. In short, when the `source` validator `withdrawable_epoch` is reached, the slashability of the `source` stake is instantly transferred to the `target` validator (see figure above).

Why do consolidating validators need to have the same withdrawal credential?

Exactly as in the case of a withdrawal, it is not safe to authenticate a `Consolidation` through the BLS staking key, because it is only the withdrawal credential that should have ownership of the funds.

Unfortunately, the CL is not currently able to check EL signatures, so being able to process `Consolidation` operations with arbitrary `source` and `target` would require either:

- Introducing the secp256k1 curve in the Consensus Layer
- Having `Consolidation` operations be processed by the Execution Layer, similarly to EL-triggered withdrawals.

Either option widens the scope of this feature significantly, bringing extra complexity to an EIP that already has a lot of moving parts and some cross-layer interactions.

Therefore, we think it is best to make the consolidation feature more limited, by restricting it to `source` and `target` validators which have the same withdrawal credential. No signature from the withdrawal credentials is needed in this case, because the `Consolidation` does not change the ultimate owner of the funds. We can then just authenticate the operation by checking a signature by the staking key of the `source` validator, as we would for an exit. From discussions with large node operators, this doesn't seem to be a severe limitation. Lido for example, only has a single withdrawal credential for all of their validators across different node operators. Many other staking operations reuse withdrawal credentials, making the consolidation feasible for them as well.

Why can we only consolidate two validators at a time?

- To simplify the design, the consolidation operation itself only contains two validators. However, many `source` validators can be concurrently merged into the same `target` (e.g., concurrent pairwise merging). For example, a node operator with a single `target` and can have two source validators `sourceA` & `sourceB` consolidating into it at the same time. The only constraint on this rate is due to the churn limit, there to preserves a sufficiently long weak subjectivity period, as we discuss later.

How long does a single pairwise consolidation take?

- The minimum consolidation time is the ~27 hours of the `MIN_WITH_VALIDATOR_WITHDRAWABILITY_DELAY`, and on top of that a `Consolidation` takes as long as the exit of the `source` validator. Like all the rate-limited queues in the consensus layer, this depends on the amount of usage in the queue at a given time. The churn limit for `Consolidation` operations is set to 256 `ETH` per epoch, equivalent to 8 validators of today. See the following set of questions for more details on the churn limit, weak subjectivity, and rate at which consolidation can occur.

Consolidations and churn limit

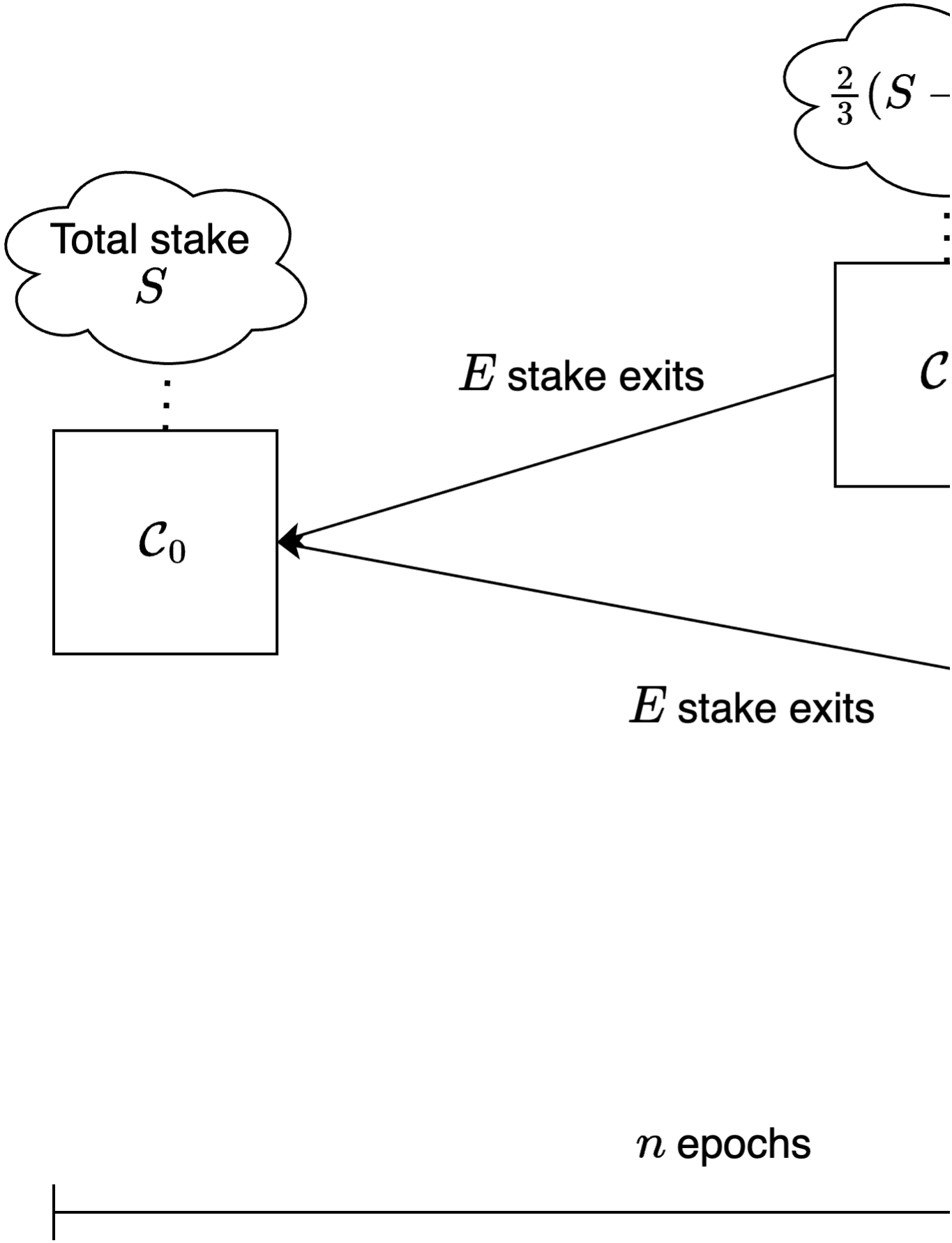
Define accountable safety and the weak subjectivity period.

Accountable safety is the guarantee that two conflicting finalized checkpoints lead to the identification and slashing of at least $\frac{1}{3}$ of the stake. We get this from a simple quorum intersection argument: since we require a $\frac{2}{3}$ quorum to finalize, any two quorums must intersect in at least $\frac{1}{3}$ of the stake. In the context of a dynamic validator set, however, the quorum intersection property, and thus accountable safety, deteriorate over time, as more and more of the validator set is allowed to change. The guarantee that we have is instead of the following form: if two conflicting checkpoints are finalized within n epochs of their highest common ancestor $\mathcal{C}$, then we can identify and slash $\frac{1}{3} - D(n)$ of the stake in $\mathcal{C}$, for an increasing function $D(n) \in [0,1]$ depending on the churn limits we set on activations and exits per epoch. For a desired maximum tolerable safety deterioration $D_0 \in [0, \frac{1}{3})$, the weak subjectivity period is the minimum n such that $D(n) \ge D_0$, i.e., the minimum number of epochs such that a conflicting finalization can happen without at least a $\frac{1}{3} - D_0$ of the stake being slashable.

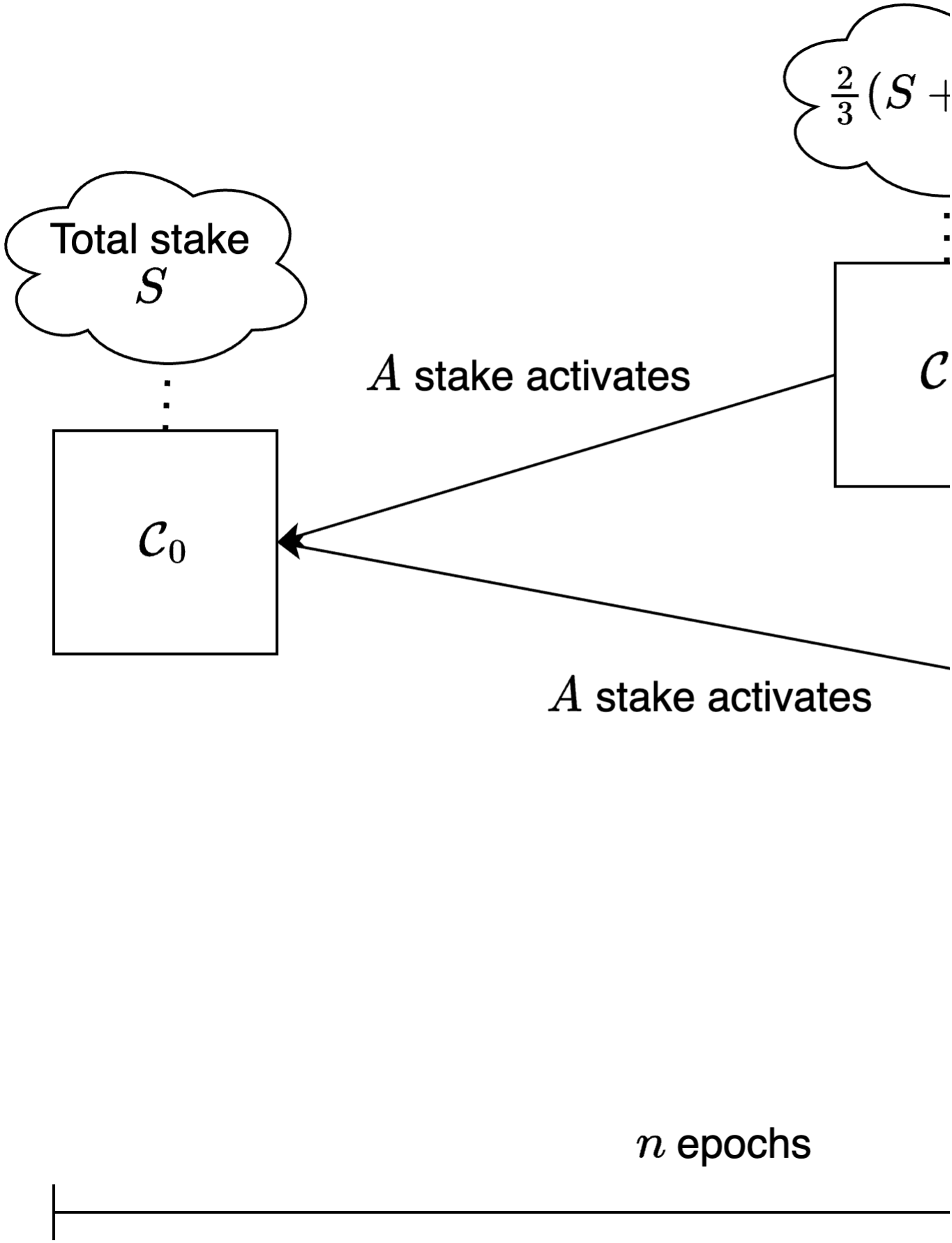
How do exits and activations affect the weak subjectivity period?

Say we have two branches with diverging validator sets of total stake S_1 and S_2, and total stake S *across the two branches*. Here, S can be less than $S_1 + S_2$, if there still is overlap between the validator sets. Then, the minimum intersection of quora for the two branches (i.e. the minimum slashable stake in case of conflicting finalizations produced by these validator sets) is $\frac{2}{3}S_1 + \frac{2}{3}S_2 - S$, i.e., the sum of the quorum sizes minus the total stake.

Let's now use this to understand the effect of exits. In the figure, we start with a checkpoint $\mathcal{C}_0$ where the total stake is S, and throughout n epochs the exit of E stake is allowed. In two chains branching from $\mathcal{C}_0$, *different* sets of validators with cumulative stake E are exited, and eventually conflicting checkpoints $\mathcal{C}_1$ and $\mathcal{C}_2$ are finalized. Due to the exits, the required quorum on either branch is now only $\frac{2}{3}(S - E)$, while the total stake across the two branches is still S, because all validators are still active on at least one of the two branches. Therefore, the minimum intersection between the quorum for $\mathcal{C}_1$ and the one for $\mathcal{C}_2$ is $2*\frac{2}{3}(S-E) - S = \frac{1}{3}S - \frac{4}{3}E$, corresponding to a safety loss of $\frac{4}{3}E$. Another way to think about this is that an exit of E stake reduces the safety by $\frac{2}{3}E$ because it reduces the required quorum by that amount, and enacting different exits on two branches doubles the safety loss.



Let's do the same analysis for activations. Everything is as before, except instead of exits we have the activation of A stake on either branch. Intuitively, the activation of A stake should reduce safety by $A - \frac{2}{3}A = \frac{1}{3}A$. This is because the activating stake is free to vote for whatever it wants without being slashed, but it increases the required quorum by $\frac{2}{3}A$. Enacting different activations on two branches should again double the safety loss, for a total of $\frac{2}{3}A$. In fact, either branch now has a quorum of $\frac{2}{3}(S+A)$ and the total stake across the branches is $S + 2A$, so the minimum quorum intersection between the two is $2*\frac{2}{3}(S + A) - (S + 2A) = \frac{1}{3}S - \frac{2}{3}A$.



The effect of activations and exits (involving different validators) simply adds up: allowing activation of A stake and exit of E stake makes the quorum $\frac{2}{3}(S + A - E)$ and the total stake across both branches $S + 2A$, so the minimum quorum intersection is $\frac{4}{3}(S + A - E) - (S + 2A) = \frac{1}{3}S - \frac{2}{3}A - \frac{4}{3}E$.

With this, we can now compute the weak subjectivity period with a symmetric churn limit of d stake per epoch both for activations and exits, and an initial total stake of S. After n epochs, the minimum quorum intersection reduces to $\frac{S}{3} - \frac{2}{3}(n*d) + \frac{4}{3}(n*d) = \frac{S}{3} - 2n*d$, so the safety degradation is $D(n) = \frac{2n*d}{S}$. The weak subjectivity period for maximum tolerable safety loss D_0 is then $n = \frac{D_0S}{2d}$. You find this same expression in this document, modulo the churn being expressed in validators per epoch rather than stake per epoch.

How do consolidations affect the weak subjectivity period?

A consolidation is equivalent to an exit followed by activation, so we would expect allowing consolidation of C stake to cause a safety reduction of $\frac{2}{3}C + \frac{4}{3}C = 2C$, i.e., the effect of allowing exits and activations of C stake. A single consolidation of C stake causes a safety loss of exactly C because the quorum is unchanged (we are only moving stake around, not changing the total amount), but the consolidated stake can now vote for whatever it wants, as with an activation. With consolidations in two different branches, we then get precisely to the expected safety reduction of $2C$. The consolidating stake is essentially free to jump from one quorum to a conflicting one without counting in the intersection: the consolidation frees it from any previous voting commitments made by the source validator, protecting it from slashing. To see this more concretely, just evenly divide the initial stake S between two sets of validators V_1 and V_2. In one branch, the C stake consolidates from V_1 to V_2, and in another chain, the opposite happens. In the first branch, V_2 has $\frac{S}{2} + C$ stake, so it only needs $\frac{2}{3}S - (\frac{S}{2} + C)$ stake from V_1 to reach a quorum and finalize, and the analog holds for V_1 in the second branch. Therefore, a double finalization only requires $\frac{4}{3}S - (S + 2C) = \frac{1}{3}S - 2C$ stake to be slashable. Were we to set the churn limit *for consolidations* at d_C stake, the weak subjectivity period would then be $\frac{D_0S}{2(d + d_C)}$, equivalent to the effect of increasing the churn limit for both activations and exits by d_C.

How do we avoid a reduction in the weak subjectivity period?

In the Beacon Chain the churn limit is not a fixed amount of stake, and instead depends on the total stake S. Up to the Deneb hard fork, it has been set to $\max(4, \frac{N}{2^{16}})$ *validators*, corresponding to $d(S) = \max(128, \frac{S}{2^{16}})$ stake in a post-MaxEB spec, where validators can have variable balance and the churn is measured in stake rather than validators. In Deneb, EIP-7514 introduces the MAX_PER_EPOCH_CHURN_LIMIT parameter, which bounds the *activation* churn to 8 validators. Incidentally, this has the effect of increasing the weak subjectivity period whenever the bound is active, i.e. whenever the number of validators is $> 2^{19}$, or in a post-MaxEB spec whenever the total stake is $> 2^{24}$. We propose to restore the symmetry of exits and activations by applying the same bound to the exit churn, and, whenever the total stake is $> 2^{24}$ and there is a "leftover" churn, to use that for consolidations. For example, if the total stake is 2^{25}, the exit and activation churn would originally each have been set to 16 validators, equivalent to 512 ETH. The bound leaves a "weak subjectivity budget" of 256 ETH worth of exits and 256 ETH worth of activations, which correspond to 256 ETH worth of consolidations, and we set the consolidation churn limit precisely to this. In other words, we set the churn limit for activations and exits to $d_{AE}(S) = \min(256, d(S))$, and the churn limit for consolidations to $d_C(S) = d(S) - d_{AE}(S)$, so that the weak subjectivity period remains $\frac{D_0S}{2(d_{AE}(S) + d_C(S))} = \frac{D_0S}{2d(S)}$. Bounding the activation and exit churn can be just a temporary measure to allow for consolidations to happen, and we can eventually revert to the original churn limit and deprecate consolidations.

How long would it take for the validator set to consolidate?

This depends on the total stake S. If $S \le 2^{24}$, $d_C(S) = 0$ in our current proposal, so there cannot be any consolidation *through this feature*, though it is in principle still possible for consolidation to happen just through exits and activations. On the other hand, such a dramatic decrease in the amount staked is unforeseeable in the short term. With the current total stake of ~29M ETH, the consolidation churn would be $29\frac{2^{20}}{2^{16}} -256 = 208$ ETH per epoch. Consolidations do not affect the total amount of stake and therefore do not affect the churn limit, so consolidation of *all* validators would require at least $29M/208$ epochs, about ~21 months, assuming no significant net inflow from exit and activations affects the churn limit during the process either. At 32M staked, the consolidation churn would be 256 ETH per epoch and full consolidation would require at least ~19 months.