---

title: Block proposal description: Explanation of how blocks are proposed in proof-of-stake Ethereum. lang: en

---

Blocks are the fundamental units of the blockchain. Blocks are discrete units of information that get passed between nodes, agreed upon and added to each node's database. This page explains how they are produced.

Prerequisites {#prerequisites}

Block proposal is part of the proof-of-stake protocol. To help understand this page, we recommend you read about proof-of-stake and block architecture.

Who produces blocks? {#who-produces-blocks}

Validator accounts propose blocks. Validator accounts are managed by node operators who run validator software as part of their execution and consensus clients and have deposited at least 32 ETH into the deposit contract. However, each validator is only occasionally responsible for proposing a block. Ethereum measures time in slots and epochs. Each slot is twelve seconds, and 32 slots (6.4 minutes) make up an epoch. Every slot is an opportunity to add a new block on Ethereum.

Random selection {#random-selection}

A single validator is pseudo-randomly chosen to propose a block in each slot. There is no such thing as true randomness in a blockchain because if each node generated genuinely random numbers, they couldn't come to consensus. Instead, the aim is to make the validator selection process unpredictable. The randomness is achieved on Ethereum using an algorithm called RANDAO that mixes a hash from the block proposer with a seed that gets updated every block. This value is used to select a specific validator from the total validator set. The validator selection is fixed two epochs in advance as a way to protect against certain kinds of seed manipulation.

Although validators add to RANDAO in each slot, the global RANDAO value is only updated once per epoch. To compute the index of the next block proposer, the RANDAO value is mixed with the slot number to give a unique value in each slot. The probability of an individual validator being selected is not simply `1/N` (where `N` = total active validators). Instead, it is weighted by the effective ETH balance of each validator. The maximum effective balance is 32 ETH (this means that `balance < 32 ETH` leads to a lower weight than `balance == 32 ETH`, but `balance > 32 ETH` does not lead to higher weighting than `balance == 32 ETH`).

Only one block proposer is selected in each slot. Under normal conditions, a single block producer creates and releases a single block in their dedicated slot. Creating two blocks for the same slot is a slashable offence, often known as "equivocation".

How is the block created? {#how-is-a-block-created}

The block proposer is expected to broadcast a signed beacon block that builds on top of the most recent head of the chain according to the view of their own locally-run fork choice algorithm. The fork choice algorithm applies any queued attestations left over from the previous slot, then finds the block with the greatest accumulated weight of attestations in its history. That block is the parent of the new block created by the proposer.

The block proposer creates a block by collecting data from its own local database and view of the chain. The contents of the block are shown in the snippet below:

```
rust class BeaconBlockBody(Container): randao_reveal: BLSSignature eth1_data: Eth1Data graffiti: Bytes32
proposer_slashings: List[ProposerSlashing, MAX_PROPOSER_SLASHINGS] attester_slashings: List[AttesterSlashing,
MAX_ATTESTER_SLASHINGS] attestations: List[Attestation, MAX_ATTESTATIONS] deposits: List[Deposit, MAX_DEPOSITS]
voluntary_exits: List[SignedVoluntaryExit, MAX_VOLUNTARY_EXITS] sync_aggregate: SyncAggregate execution_payload:
ExecutionPayload
```

The `randao_reveal` field takes a verifiable random value that the block proposer creates by signing the current epoch number. `eth1_data` is a vote for the block proposer's view of the deposit contract, including the root of the deposit Merkle trie and the total number of deposits that enable new deposits to be verified. `graffiti` is an optional field that can be used to add a message to the block. `proposer_slashings` and `attester_slashings` are fields that contain proof that certain validators have committed slashable offenses according to the proposer's view of the chain. `deposits` is a list of new validator deposits that the block proposer is aware of, and `voluntary_exits` is a list of validators that wish to exit that the block proposer has heard about on the consensus layer gossip network. The `sync_aggregate` is a vector showing which validators were previously assigned to a sync committee (a subset of validators that serve light client data) and participated in signing data.

The `execution_payload` enables information about transactions to be passed between the execution and consensus clients. The `execution_payload` is a block of execution data that gets nested inside a beacon block. The fields inside the `execution_payload` reflect the block structure outlined in the Ethereum yellow paper, except that there are no ommers and `prev_randao` exists in place of `difficulty`. The execution client has access to a local pool of transactions that it has heard about on its own gossip network. These transactions are executed locally to generate an updated state trie known as a post-state. The transactions are included in the `execution_payload` as a list called `transactions` and the post-state is provided in the `state-root` field.

All of these data are collected in a beacon block, signed, and broadcast to the block proposer's peers, who propagate it on to their peers, etc.

Read more about the anatomy of blocks.

What happens to the block? {#what-happens-to-blocks}

The block is added to the block proposer's local database and broadcast to peers over the consensus layer gossip network. When a validator receives the block, it verifies the data inside it, including checking that the block has the correct parent, corresponds to the correct slot, that the proposer index is the expected one, that the RANDAO reveal is valid and that the proposer is not slashed. The `execution_payload` is unbundled, and the validator's execution client re-executes the transactions in the list to check the proposed state change. Assuming the block passes all these checks, each validator adds the block to its own canonical chain. The process then starts again in the next slot.

Block rewards {#block-rewards}

The block proposer receives payment for their work. There is a `base_reward` calculated as a function of the number of active validators and their effective balances. The block proposer then receives a fraction of `base_reward` for every valid attestation included in the block; the more validators attest to the block, the greater the block proposer's reward. There is also a reward for reporting validators that should be slashed, equal to `1/512 * effective balance` for each slashed validator.

More on rewards and penalties

Further reading {#further-reading}

- Introduction to blocks
- Introduction to proof-of-stake
- Ethereum consensus specs
- Introduction to Gasper
- Upgrading Ethereum