The WhitePaper Reading Club Kuala Lumpur [02] [29/3/2025]

Ethereum: Pectra Upgrade

Rongxin, Yudhishthra, Yee Chian, Aik Wei, Justin

1 Liner: Explore how Ethereum's Pectra upgrade enhances staking and boosts validator efficiency.

EIPS: (1) EIP-2537: Precompile for BLS12-381 curve operations **(2)** EIP-2935: Save historical block hashes in state **(3)** EIP-6110: Supply validator deposits on chain **(4)** EIP-7002: Execution layer triggerable withdrawals **(5)** EIP-7251: Increase the MAX_EFFECTIVE_BALANCE **(6)** EIP-7549: Move committee index outside Attestation **(7)** EIP-7623: Increase calldata cost **(8)** EIP-7685: General purpose execution layer requests **(9)** EIP-7691: Blob throughput increase **(10)** EIP-7702: Set EOA account code **(11)** EIP-7840: Add blob schedule to EL config files

Why this is important: EIP-7251, EIP-6110, EIP-7002, and EIP-7549 collectively enhance Ethereum's PoS by improving staking flexibility and rewards for users, streamlining validator activation and management for stakeholders, and boosting network efficiency and developer capabilities for partners.

Key Innovation: EIP-7251: Increase the MAX_EFFECTIVE_BALANCE, EIP-6110: Supply Validator Deposits on Chain, EIP-7002: Execution Layer Triggerable Exits, EIP-7549: Move committee index outside Attestation

Overview: The Pectra upgrade enhances Ethereum's Proof of Stake (PoS) system by introducing key improvements through Ethereum Improvement Proposals (EIPs) designed to make staking more flexible, efficient, and user-friendly. EIP-7251 increases the maximum validator balance from 32 ETH to 2,048 ETH, allowing stakers to commit larger amounts and reducing the total number of validators needed, which streamlines network operations. EIP-6110 speeds up validator activation by embedding deposit data directly into blocks, cutting wait times from over 48 hours to just minutes. EIP-7002 gives stakers more control by letting them trigger exits from the Execution Layer, improving autonomy, especially for delegated staking, while EIP-7549 lowers the cost of verifying signatures, boosting overall network efficiency. These upgrades mean stakers enjoy higher rewards and faster participation, app developers can build better staking tools, and users might see lower fees and quicker transactions—though some worry about centralization risks and balancing innovation with stability.

Questions:: (i) Threats/Attacks in Pectra? Security Audit for Pectra

Team: **Mikhail Kalinin** is the Lead Researcher at ConsenSys R&D and a researcher at the Ethereum Foundation. He contributed to the development of the **Teku** client (an Ethereum 2.0 client) and designed mechanisms for Ethereum's "The Merge." He is one of the authors of EIP-6110, which aims to simplify processes and improve efficiency by handling validator deposits on-chain. **Danny Ryan (Danny Ryan – Medium)** was a researcher at the Ethereum Foundation, where he coordinated network upgrades, including the 2021 launch of the Beacon Chain, a pivotal step in Ethereum's shift to Proof of Stake (PoS). He contributed to several EIPs, such as EIP-1559, which made transaction fees more predictable. He recently left the Ethereum Foundation, but his impact on Ethereum's development remains profound. **Dapplion** is a core developer for **Lighthouse**, an Ethereum Beacon Chain client implementation. He also serves as the Merge Coordinator for Gnosis Chain and previously worked on ChainSafe's **Lodestar** project. He has made significant contributions to Ethereum's scalability and validator management, actively participating in various EIP discussions.

Opinions: It feels a bit light on details for those who want to dig deeper. For example, it mentions EIP-7251 allows staking up to 2048 ETH, but doesn't explore how this might lead to more centralization, which is a big concern in the community. Also, it could connect these changes to Ethereum's bigger picture, like future scalability plans. Still, for most users, it's a helpful guide, balancing technical info with accessibility.

Conceptual Components of EIP-7251:

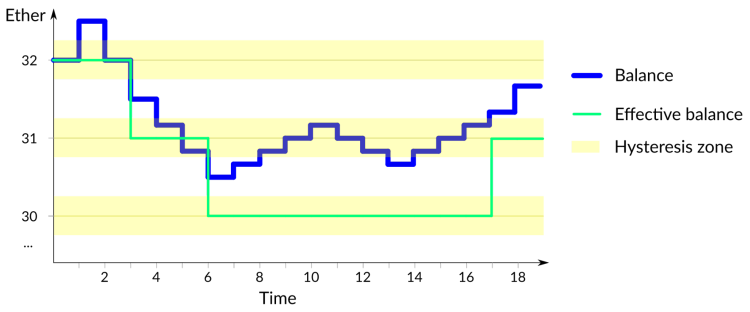
Problems: As of October 3, 2023, over 830,000 validators are active in the consensus layer, a number growing due tol the 32 ETH cap set by max effective balance. This creates "redundant validators"—multiple units under one entity, often on the same beacon node, using distinct BLS keys.

Solution: Increases the constant max effective balance, while keeping the minimum staking balance 32 ETH

Effective balance	A value for each validator that is derived from the validator's balance and prior effective balance. It is used to calculate the size of rewards and penalties given to that validator.

(i) Effective balance can never be more than 32Ξ. This means, for example, that if a validator's balance is 100Ξ its effective balance will be 32Ξ **(ii)** Effective balance is always a multiple of 1Ξ, with any additional balance being ignored. This means, for example, that if a validator's balance is 29.7Ξ its effective balance will be 29Ξ, with the extra 0.7Ξ ignored for reward and penalty calculations **(iii)** Effective balance will only increase if the validator's balance is more than 1.25Ξ higher than its current effective balance. This means, for example, that if a validator's effective balance is 25Ξ its balance must increase to more than 26.25Ξ before its effective balance will increase to 26Ξ **(iv)** Effective balance will only decrease if the validator's balance is more than 0.25Ξ lower than its current effective balance. This means, for example, that if a validator's effective balance is 25Ξ its balance must decrease to less than 24.75Ξ before its effective balance will decrease to 24Ξ

Following the above rules allows the addition of a line showing the validator's effective balance to the chart:



Content of upgrades

EIP-7251 raises this cap on a validator's effective balance to 2048 ETH, making it possible to stake any amount of ETH between 32 and 2048 ETH in a single validator. Further, it exempts validators who opt-in from the automated "sweep" process, so these validators' balances can grow in excess of 32 ETH. Importantly, validators who opt-in to EIP-7251 can still manually trigger partial withdrawals, but this will no longer happen automatically.

Opting in to this feature is not required, and any validators who prefer the current system can continue staking in 32 ETH increments without any change. Operationally, adopting this feature and increasing a validator's max effective balance requires updating the validator's withdrawal credential from either the 0x00 or 0x01-type withdrawal credentials, to a new 0x02-type withdrawal credential.

Why is it important? (i) Enhances network efficiency: Reduces the number of validators, optimizing communication and consensus processes. **(ii) Improves user rewards:** Supports compounding mode and flexible withdrawals. **(iii) Simplifies management**: Consolidates validators, lowering the operational threshold.

4.Supports scalability: Facilitates staking pools and institutional participation, driving ETH staking growth.

5.Strengthens security: Concentrates voting weight, accelerating finalization.

It addresses the core issues of validator bloat and reward limitations, making Ethereum's PoS system more efficient, flexible, and sustainable, laying the groundwork for staking millions more ETH in the future.

What it means

1.For Stakers (i)Higher Rewards: MaxEB rises to 2048 ETH, enabling compounding rewards for larger stakes. (ii) Flexible Withdrawals: With EIP-7002, stakers can manually withdraw specific amounts (e.g., 10 ETH), adapting to market needs. (iii)Lower Costs: Validator consolidation (e.g., 96 ETH in 1 validator) cuts hardware and bandwidth expenses, aiding solo stakers. (iv)Small Stakers: Minimum remains 32 ETH, but manual withdrawals add Gas costs.

2. For App Builders (i) Flexible dApps: Build staking tools with compounding options and withdrawal interfaces, boosting user appeal. (ii) Pool Efficiency: Optimize contracts (e.g., Lido) to lower costs and pass savings to users. (iii) Challenge: Updating dApps for new modes (e.g., 0x02 credentials) increases workload.

3. For Users (i) Lower Fees: Better network efficiency may cut Gas costs during peak times. (ii) Stable Network: Faster, more reliable transactions (12s confirmation, 6.4m finalization).

Question: **(i)** Does this increase the cost of validation or efficiency of validation? **(ii)** Does this increase the validation set? Does this mean there needs to be less validators or the same amount? **(iii)** Proposer vs Validator affect the chances of you getting accepted as a validator or prosper.

Security Audit

Medium Severity Issue M1: Fee Update Logic Can Lead to Denial of Service (DoS):The report states that the excess_request_count is only updated at the end of each block via the system path (page 8), not during user requests in the user path (page 6). This allows an attacker to submit multiple consolidation requests within a single block at a fixed fee, inflating the fee for the next block and rendering the contract unusable for hours—a DoS attack.

Relevant Code User Path:

```
if (96 == msg.data.length) {
    require(msg.value >= v15 / 17);  // Fee check
    request_count += 1;              // Increment request
counter// Store request data (msg.sender and pubkeys)
    STORAGE[4 + 4 * queue_tail_index] = msg.sender;
    STORAGE[5 + 4 * queue_tail_index] = calldata_0_32;
    STORAGE[6 + 4 * queue_tail_index] = calldata_32_64;
    STORAGE[7 + 4 * queue_tail_index] = calldata_64_96;
    queue_tail_index += 1;           // Increment tail pointer
}
```

Note: excess_request_count is not updated here; the fee (v15 / 17) is based on the prior value.

System Path

```
v5 = v6 = excess_request_count;
if (uint256.max == v6) {
    v5 = 0;
}
if (request_count + v5 > 1) {
    v8 = request_count + v5 - 1;
} else {
    v8 = 0;
}
excess_request_count = v8;  // Updated only here, at block end
request_count = 0;          // Reset request_count
```

Fee Calculation

```
v11 = v12 = 17;
v13 = v14 = 1;
v15 = v16 = 0;
while (v11 > 0) {
    v15 += v11;
    v11 = excess_request_count * v11 / (v13 * 17);
    v13 += 1;
}
// Result: v15 / 17 as the fee
```

Code Analysis (i) Root Cause: In the user path, add_consolidation_request increments request_count and stores data but does not update excess_request_count. Within a single block, multiple calls use the same excess_request_count, keeping the fee static. (ii)Attack Vector:(a) An attacker loops add_consolidation_request in one transaction, submitting 780 (new slots) or 1650 (reused slots) requests.(b)With excess_request_count unchanged, the fee remains low (e.g., 0.000112 ETH

	at 550 requests).(c) At block end, the system path updates excess_request_count, causing the next block's fee to spike (e.g., 273 ETH at 800 requests), making the contract unusable. **Impact:** Low gas cost (optimized with zero calldata) enables a 2.6-5.5 hour DoS.

Conceptual Components of EIP-6110:

Motivation: Validator deposits are a core component of the proof-of-stake consensus mechanism. This EIP allows for an in-protocol mechanism of deposit processing on the Consensus Layer and eliminates the proposer voting mechanism utilized currently.

Content of upgrades	**Current Process and Issues:**To activate a validator in Ethereum's PoS system, 32 ETH is deposited into the Ethereum Deposit Contract on the Execution Layer (EL). This deposit data must then reach the Consensus Layer (CL) to register the validator on the Beacon Chain. Currently:**(i) Eth1Data Voting Mechanism**: CL validators vote on EL state (e.g., block hash and deposit count) using the Eth1Data process. A majority vote confirms the data, but a "Follow Distance" of ~2048 EL blocks (~8 hours) delays this for security.**(ii) Delays and Complexity**: The process takes 48+ hours due to voting and network conditions. Block proposers must manually track deposits, adding complexity and error risk, especially with 830,000 validators (October 2023). EIP-6110 Improvements:EIP-6110 streamlines this by embedding deposit data directly into EL blocks, eliminating voting. Key changes:(i) On-Chain Deposit Data: EL clients (e.g., Geth) extract deposit events and add them to a new "Deposit Requests" field in each block.(ii) Direct Transfer to CL: The CL automatically parses these requests from EL blocks, adding validators to the registry without voting.(iii) Faster Activation: New validators join the Activation Queue in minutes, not 48 hours, limited only by queue processing speed. // **Question:** Why does it matter how fast the staker can join? Why doesn't it matter how fast they can leave? E.g healthy banks can handle large bank runs. Benefits:(i)Speed: Activation time drops from 48 hours to minutes.(ii)Simplicity: No manual tracking or voting complexity.(iii)Efficiency: Scales better with growing validator numbers. EIP-6110 makes staking faster and simpler by integrating deposit data on-chain.
What it means	1) 1. For Stakers:(i) Faster Activation: Validator setup drops from 48 hours to minutes, starting rewards sooner.(ii)Simpler Process: No need to track Eth1Data votes, reducing complexity.(iii) Predictability: Consistent, quick activation timelines. 2) 2. For App Builders:(i) Better Onboarding: Near-instant staking improves dApp user experience.(ii) Easier Integration: Deposit data in blocks simplifies development.(iii) New Tools: Real-time staking features enhance offerings. 3) 3. For Users:(i) Quick Staking: Rewards start faster via pools or wallets.(ii) Network Efficiency: Lower load may reduce Gas fees.(iii) Increased Trust: Streamlined process encourages staking.

Security Audit	1. Medium-Severity Issue (M1) - Fee Update Logic Can Lead to DoS. Description: (i) The excess_request_count (excess request counter) is only updated once per block during the system call, not after each user-submitted request. (ii) This allows an attacker to submit multiple "bogus" requests within the same block at a fixed fee, inflating the fee for subsequent blocks and rendering the contract unusable for hours (a Denial of Service, or DoS, attack).
	Relevant Code
	`// update_excess_withdrawal_requests in the system path`
	`v6 = v7 = excess_request_count;`
	`if (uint256.max == v7) {`
	`  v6 = 0;`
	`}`
	`if (request_count + v6 > 2) {`
	`  v9 = request_count + v6 - 2;`
	`} else {`
	`  v9 = 0;`
	`}`
	`excess_request_count = v9; // Updated only at block end`

The issue is that excess_request_count isn't updated immediately in the user path after each request.
Impact:Attackers can flood the queue with low-fee requests, spiking excess_request_count and causing fees to surge, making the contract unusable.
Suggested Fix:Update excess_request_count immediately after each request in the user path, e.g.:

```
// In add_consolidation_request
request_count += 1;
excess_request_count = calculate_new_excess(request_count, excess_request_count);
```

Status: Unresolved (OPEN); Dedaub considers the fix simple and necessary.
2. Advisory Issue (A1) - Fee Calculation Can Be OptimizedDescription:(i) The contract recalculates the fee (fake_exponential function) for every request in the user path, even though the fee remains constant within a block.(ii) This causes unnecessary gas consumption, as system-side calculations are "free" (borne by the system).
Relevant Code:

```
// fake_exponential called per request in user path
v12 = 17;
v14 = 1;
v16 = 0;
while (v12 > 0) {
    v16 += v12;
    v12 = excess_request_count * v12 / (v14 * 17);
    v14 += 1;
}
require(msg.value >= v16 / 17); // Recalculated every time
```

Impact:Repeated fee calculations increase gas costs, particularly in high-frequency request scenarios.
Suggested Fix:Pre-calculate the fee in the system path at the start of each block and store it for user path reuse (memoization-like approach), e.g.:

```
// In system path
STORAGE[some_slot] = v16 / 17; // Store fee
// In user path
require(msg.value >= STORAGE[some_slot]);
```

Status: Advisory (INFO); not a mandatory fix.
3.Advisory Issue (A2) - Constant Operation Can Be Optimized Description:In the fake_exponential function, a multiplication between two constants (WITHDRAWAL_REQUEST_FEE_UPDATE_FRACTION and MIN_WITHDRAWAL_REQUEST_FEE) is performed dynamically, though it could be precomputed to save runtime gas.
Relevant Code:

```
; fake exponentiation
dup3          ; [denom, factor, numer, denom]
mul           ; [accum, numer, denom]  // Constant multiplication
```

Here, factor and numerator are constants whose product could be precalculated.
Impact:Each call performs an unnecessary multiplication, adding minor gas overhead.
Suggested Fix:Hardcode the precomputed result, e.g.:

```
push PRECALCULATED_VALUE  ; Push precomputed result directly
```

Status: Advisory (INFO); not a mandatory fix.

Conceptual Components of EIP-7002:

Motivation: EIP-7002 allows validator exits to be triggered from Ethereum's Execution Layer (EL), giving stakers—especially those who delegate the operation of their validator to a third-party—greater control over when their validator exits the network. Additionally, this EIP provides validators who opt-in to EIP-7251 the ability to manually trigger partial withdrawals.

Content of upgrades	1) **Core Mechanism**: Currently, only the validator's active key can initiate exits, risking fund lockup if lost or controlled by others. EIP-7002 allows 0x01 withdrawal credential holders (e.g., cold storage addresses) to trigger exits directly from the Execution Layer, bypassing the active key. 2) **Implementation**: Adds "Exit Requests" to the Execution Layer, relayed to the Consensus Layer via the Engine API, with a dynamic fee system (like EIP-1559) limiting requests (default: 2 per block) to prevent abuse. 3) **Synergy with Other EIPs**: Complements EIP-7251 (increasing max effective balance) to enhance fund management for large stakers. 4) **Importance**:**(i) Autonomy and Decentralization**: Empowers stakers, especially in delegated setups (e.g., Rocket Pool), by giving withdrawal credential holders exit control and ensuring fund recovery without the active key. Reduces reliance on centralized operators, promoting DVT and solo staking.**(ii) User Experience**: Simplifies exits (no pre-signed messages or Consensus Layer access) and, with EIP-7251, enables partial withdrawals for flexible reward extraction.**(iii) Security**: Dynamic fees deter DoS attacks, while increased staking participation raises the cost of a 51% attack.
What it means	1) 1. For Stakers (i) Greater Control: Exit using withdrawal credentials (e.g., cold wallets), bypassing active key reliance in pools.(ii) Improved Security: Recover funds if the active key is lost.(iii) Flexible Withdrawals: Partial exits (with EIP-7251) allow withdrawing rewards while staying active. 2) 2. For App Builders(i) Trustless Features: Build pools where users trigger exits directly, reducing operator dependency.(ii) Simplified UX: No pre-signed exits; integrate seamless exit requests.(iii) New Tools: Develop interfaces for partial withdrawals or exit tracking. 3) 3. For Users(i) Faster Pool Exits: Redeem ETH from pools (e.g., stETH) quicker without delays.(ii) Network Resilience: More secure staking boosts ETH locked, enhancing security.(iii) Lower Costs (Potential): Streamlined exits may reduce Gas fees. 4) 4. For Staking Pools (e.g., Lido)(i) Reduced Operator Power: Users can exit without operator cooperation, shifting control to withdrawal credential holders.(ii) Operational Shift: Pools must adapt to trustless exits, potentially integrating EIP-7002 into governance (e.g., Lido's dual governance).(iii) User Appeal: Faster, autonomous exits enhance attractiveness, but pools lose leverage over staked ETH timing.

g

Motivation: This proposal makes Casper FFG clients more efficient by reducing the average number of pairings needed to verify consensus rules.

Content of upgrades	1) **Core Change**:**(i)** Current: Beacon Chain validators' attestation messages include the Committee Index, with each committee generating a separate signature, resulting in up to 64 signatures.**(ii)** EIP-7549: Moves the Committee Index out of the signed portion, allowing all votes to aggregate into 1 signature based on identical data. 2) **Implementation**:(i) Modifies attestation message structure, treating the Committee Index as an unsigned field.(ii) Implemented via the Electra hard fork, backward-incompatible, requiring synchronized Consensus Layer updates. 3) Importance(i) Improved Consensus Efficiency: (a) Reduces signatures from 64 to 1, cutting BLS signature verification pairing costs by ~98%, optimizing node performance.(b) Benefits Casper FFG finality voting by lowering computational overhead.(ii) Support for ZK Circuit Optimization:ZK proofs (e.g., zk-SNARKs) verifying Casper FFG consensus see signature validation as a bottleneck. EIP-7549 drops this from 64 to 1, greatly boosting efficiency for light clients and off-chain validation.(iii) Enhanced Network Scalability:(a) Lowers network load from signature aggregation, accommodating surging validator counts.(b) Ensures performance for future large-scale staking.(iv) Trade-offs and Challenges:(a) Increases attestation message size (bitfield expands 64x), potentially impacting bandwidth.(b) First block post-hard fork may lack attestations, briefly reducing FFG votes, though the impact is minimal.
What it means	1) 1. For Stakers(i) Lower Operational Costs:Reduces signature aggregation from 64 to 1, cutting CPU and memory usage for validator nodes.(ii) Improved Performance:Faster attestation processing enhances efficiency, especially for solo stakers with limited hardware.

	2) 2. For App Builders(i) Optimized Light Clients:Single-signature verification simplifies ZK proofs for Casper FFG, enabling efficient light client apps.(ii) Easier Sync Tools:Reduced attestation complexity aids in building faster syncing or monitoring dApps. 3) 3. For Users(i) Network Efficiency:Lower consensus load may speed up transaction confirmations and reduce Gas fees.(ii) Better Light Client Access:Enhanced ZK proofs make lightweight apps (e.g., mobile wallets) more viable.

Politics and conflicts

Debate	1.Staking Changes (EIP-7251): This proposal raises the validator staking limit from 32 ETH to 2,048 ETH to make the network more efficient by reducing the number of validators. However, some worry this could give big players like Coinbase or Lido too much power, threatening Ethereum's goal of staying decentralized. 2.Gas Fees and Scaling: New changes (EIPs like 7691 and 7623) aim to lower gas fees and improve transaction speed, especially for layer-2 solutions. While this could make Ethereum cheaper to use, some fear it might weaken the main network, splitting activity between layers and reducing its importance. 3.Upgrade Timing: Pectra's complexity led developers to split it into two phases for safety, but this has caused disagreement. Some want all upgrades (like Verkle trees for scalability) rolled out quickly to keep Ethereum competitive, while others prefer a slower, safer approach—even if it delays key features until 2026. 4.User Experience (EIP-7702): This introduces account abstraction to make wallets more flexible and user-friendly. Supporters say it's key for mainstream adoption, but critics worry it's too complicated and could create security risks. Big Picture: These debates reflect deeper divides in Ethereum's community—between decentralization, efficiency, innovation speed, and security. Validators, developers, and users all have different priorities, making Pectra a test of how Ethereum balances these as it grows and competes with other blockchains.
Ethereum Core Devs Meeting	To assess concerns from developers, we reviewed the latest meetings before March 23, 2025. The most recent meeting, ACDC #153, held on March 20, 2025, covered topics such as Hoodi Testnet updates, Pectra Mainnet readiness, Pectra Devnet 6, history expiry, EIP-7688, and PeerDAS Devnet 5 & 6 (ACDC #153 Highlights). The discussion on history expiry highlighted a dependency on EIP-6110, with uncertainty around Pectra's mainnet launch potentially impacting the May 1, 2025, deadline for expiring pre-merge history. Earlier, ACDC #152 on March 6, 2025, focused on Holesky Testnet updates, Pectra Shadow Fork, Pectra Sepolia Fork updates, Pectra Mainnet readiness, and fee mechanism analysis (ACDC #152 Highlights). The fee mechanism analysis explored griefing risks and withdrawal processing concerns, indicating potential technical challenges that could raise developer concerns about network stability and resource allocation. ACDE #207, held on March 12, 2025, discussed Holesky finalization, Hoodi Testnet Pectra Mainnet plans, Fusaka updates, system contract error handling, RPC changes, and stateless clients (ACDE #207 Highlights). These meetings reveal a focus on resolving technical issues, with no direct mentions of developer conflicts in the summaries. A significant point of contention is the Ethereum Foundation's $165 million DeFi investment in January 2025, aimed at growing its treasury amid a 39% decline in holdings (Ethereum Foundation Moves $165 Million in ETH). This move faced community criticism for perceived disengagement and raised questions about neutrality and governance, as noted in an article from February 4, 2025, discussing leadership shakeups and scalability debates (Can Ethereum Regain Its Dominance in 2025).

Questions:

1.What strategies could smaller stakers adopt to remain competitive as larger entities leverage EIP-7251's higher MaxEB limits?

2.How do these upgrades fit into Ethereum's long-term plans, like sharding or Layer 2 growth?

3.How quickly will staking providers adopt these new features, and what might slow them down?

Reference:

1. Understanding Ethereum's Pectra upgrade | Consensys
2. https://eips.ethereum.org/EIPS/eip-7251#abstract
3. Understanding Validator Effective Balance | Attestant
4. The Shanghai/Capella Upgrade | Your Ultimate Guide to ETH Staking Withdrawals
5. EIP-6110: Supply validator deposits on chain
6. EIP-7002: Execution layer triggerable withdrawals
7. https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/keys/#validator-key
8. Dual Governance: An Overview
9. EIP-7549: Move committee index outside Attestation

CHEAT SHEET

EIP-2537: BLS12-381 Curve Operations Precompile

Definition: EIP-2537 introduces a new *precompile* (built-in function) for the BLS12-381 elliptic curve, which is used in advanced cryptography. In simple terms, it provides an efficient way for Ethereum to handle BLS12-381 operations like signature verification directly in the protocol.

Relevance to Pectra: By including this in the Pectra upgrade, Ethereum greatly reduces the cost and complexity of verifying BLS signatures and zkSNARK proofs on-chain (Ethereum Pectra Upgrade: Key Improvements and Impact | QuickNode Guides). This means decentralized applications (like those using aggregated signatures or zero-knowledge proofs) can run more efficiently and cheaply after Pectra. Developers get a faster "built-in calculator" for these heavy cryptographic tasks instead of relying on slow smart contract code.

Analogy: Think of it like adding a specialized math co-processor to a computer. Before, smart contracts doing BLS signature checks had to do all the complex math step-by-step (slow and costly in gas). With EIP-2537, there's now a dedicated engine under the hood that can do those calculations instantly, just as a calculator speeds up solving a hard equation. This *upgrade* makes tasks like multi-signature verification (for example, checking many validator signatures at once) far more practical on Ethereum.

Historical Context & Problem Solved: Previously, Ethereum only had precompiles for older curves (like BN254) which offered about 80 bits of security, whereas BLS12-381 offers ~120 bits of security (EIP-2537: Precompile for BLS12-381 curve operations). The lack of a BLS12-381 precompile meant projects either avoided using stronger cryptography or incurred high gas costs. This EIP had been proposed since 2020 but wasn't implemented until now. By finally adding it in Pectra, Ethereum addresses a long-standing need for stronger, efficient crypto operations, paving the way for more secure and advanced applications (like improving consensus signature checks or enabling new zero-knowledge use cases). (Ethereum Pectra Upgrade: Key Improvements and Impact | QuickNode Guides)

EIP-2935: Historical Block Hash Access

Definition: EIP-2935 expands Ethereum's ability to access past block hashes by storing recent historical hashes in state. Concretely, it saves the hashes of the last 8,192 blocks (about 27.3 hours of blocks) in a special system smart contract (Ethereum Pectra Upgrade: Key Improvements and Impact | QuickNode Guides). The existing `BLOCKHASH` opcode still only returns hashes for the past 256 blocks, but now older hashes (up to that 8,192 limit) can be retrieved from this contract if needed.

Relevance to Pectra: This change in the Pectra upgrade is aimed at supporting *stateless clients* and Layer-2 protocols. By having more block history available on-chain, technologies like rollups and cross-chain bridges can directly query past block data without relying on off-chain sources (Ethereum Pectra Upgrade: Key Improvements and Impact | QuickNode Guides). In other words, Ethereum becomes more self-sufficient in providing its recent history. This improves reliability for applications that need to prove or use data from an hour or two in the past (beyond the old ~15 minute window of 256 blocks).

Analogy: It's like extending the blockchain's "memory." Imagine if a security camera could only remember footage from the last few minutes unless someone manually archived it. EIP-2935 is like installing a bigger memory card in the camera – now it automatically keeps the last day's footage accessible. For Ethereum, this means a smart contract can check what happened a few hours ago (e.g. a block hash from that time) on its own, instead of trusting an external archive. This is particularly useful for layer-2 systems that need to verify older mainnet events.

Historical Context & Problem Solved: Early Ethereum assumed full nodes would always have recent blocks, limiting on-chain block lookups to 256 blocks for efficiency. However, with plans for *stateless Ethereum* (where nodes may not store all history by default) and the rise of complex layer-2 solutions, this limitation became a problem. EIP-2935 (originally proposed in 2020) addresses this by making recent block hashes part of the state that can be included in fraud proofs or rollup proofs. It solves the issue of smart contracts and light clients being blind to any block older than ~1 hour (Ethereum Pectra Upgrade: Key Improvements and Impact | QuickNode Guides), thereby enhancing trustlessness (projects no longer need to rely on third-party servers for that data) and preparing for a more stateless future.

EIP-6110: On-Chain Validator Deposits

Definition: EIP-6110 overhauls how new validator deposits are handled by moving the entire deposit processing on-chain in the Execution Layer. Instead of the Consensus Layer "voting" on deposits from the deposit contract, each Ethereum block will directly include a list of any new validator deposit operations. In essence, the Execution Layer itself gathers and passes along deposit data to the Beacon Chain in real-time (EIPs/EIPS/eip-6110.md at master · ethereum/EIPs · GitHub) (EIPs/EIPS/eip-6110.md at master · ethereum/EIPs · GitHub).

Relevance to Pectra: This change is a core part of Pectra's improvements for stakers. By appending deposits directly in each block, the wait time for a new validator to be recognized by the Beacon Chain drops from up to ~12 hours down to about 2 epochs (~13 minutes) (Ethereum Pectra Upgrade: Key Improvements and Impact | QuickNode Guides). Pectra includes EIP-6110 to streamline the onboarding of new validators and to **reduce complexity** in the consensus client software. It also eliminates a potential security weakness where the old mechanism relied on majority vote of validators to include deposits (removing that removes the chance of malicious delay or inconsistent behavior).

Analogy: Prior to this, depositing ETH to become a validator was like submitting an application that had to go through a slow committee review (with validators periodically checking an external "inbox" of deposits). EIP-6110 makes it an automated on-the-spot process – akin to instant online registration. The moment your deposit transaction is in a block, it's as if the system immediately processes your "membership," rather than waiting hours for committee approval. This is a much better user experience for someone setting up a new validator node.

Historical Context & Problem It Solves: When Ethereum transitioned to Proof of Stake (the Beacon Chain), the deposit process was a necessary but clunky bridge between the old Ethereum chain (execution layer) and the new consensus layer. It required a voting scheme (proposer votes on deposit logs) to avoid inconsistencies, which introduced delay and complexity. This was essentially *technical debt* left from the ETH1-ETH2 merge. EIP-6110 fixes this "Eth1-Eth2 bridge" issue (EIPs/EIPS/eip-6110.md at master · ethereum/EIPs · GitHub). By doing so, it improves security (an honest node can't be tricked by bad deposit votes) and reliability (no more dependency on fragile JSON-RPC polling of deposit contract events)

(EIPs/EIPS/eip-6110.md at master · ethereum/EIPs · GitHub). In short, the Pectra upgrade uses EIP-6110 to make staking more straightforward and robust: deposits are processed quickly and seamlessly as part of each block, helping new validators join the network faster and with less chance of errors.

EIP-7002: Execution Layer Triggerable Withdrawals

Definition: EIP-7002 introduces a mechanism for validators to initiate their own exits (and withdraw stake) via the Execution Layer using their withdrawal credentials. In practice, it adds a special system contract or transaction type that allows a validator (or a smart contract acting on their behalf) to request a voluntary exit or partial withdrawal by calling from the Ethereum side (using a 0x01 withdrawal address) (EIPs/EIPS/eip-7002.md at master · ethereum/EIPs · GitHub) (EIPs/EIPS/eip-7002.md at master · ethereum/EIPs · GitHub). These "withdrawal requests" are included in blocks and then honored by the Consensus Layer, meaning a validator can start the unstaking process without the validator's BLS key having to sign an exit.

Relevance to Pectra: This is a major improvement for stakers' control and flexibility. Pectra's inclusion of EIP-7002 means that if you have provided an Ethereum address as your withdrawal credential, you (or a contract you trust) can directly trigger your validator to exit or withdraw rewards on-chain (Ethereum Pectra Upgrade: Key Improvements and Impact | QuickNode Guides). Previously, only the active validator key itself could signal an exit. With this upgrade, staking services and pools can give users more autonomy: for example, a user in a staking pool could withdraw their stake without needing the pool operator to use the validator key. This reduces trust requirements and enables more **automated staking setups** (like smart contract-managed validators that can re-balance or shut down based on on-chain logic).

Analogy: Imagine you have money locked in a vault that only your hired security guard (the validator key) could open for you. If that guard went missing or refused, you were stuck. EIP-7002 is like giving you, the vault owner (with the withdrawal key), a spare mechanism to open the vault yourself. In everyday terms, it empowers the actual owner of the funds to press the "eject" button. For instance, if a staking provider runs your validator, you can still initiate an exit from your end without relying on them — similar to having a remote kill-switch for your investment.

Historical Context & Problem It Solves: After The Merge and the Shanghai upgrade (which enabled withdrawals), a pain point remained: those who staked via custodial services or smart contracts often had the withdrawal rights separate from the validator control. If the operator holding the validator key misbehaved or lost the key, the actual owner couldn't easily retrieve their ETH. Some workarounds (like pre-signed exit messages given to users) were used, but they were clumsy and risky. EIP-7002 directly addresses this by allowing the *withdrawal credential* itself to initiate exits (EIPs/EIPS/eip-7002.md at master · ethereum/EIPs · GitHub). This greatly improves the security model of staking pools and any setup where the staking key and the owner are different people. It also means lost or compromised validator keys are less catastrophic, since the owner can still withdraw using their withdrawal address. Overall, this EIP (combined with the general cross-layer request framework) modernizes Ethereum's staking UX in Pectra, making it **more trustless and user-friendly** (Ethereum Pectra Upgrade: Key Improvements and Impact | QuickNode Guides).

EIP-7251: Increase of Maximum Validator Balance

Definition: EIP-7251 raises the cap on a validator's effective staking balance from 32 ETH to **2,048 ETH**. In Ethereum's Proof of Stake, a validator's "effective balance" is the amount of stake that counts for earning rewards (and for consensus weight). Prior to this EIP, any ETH above 32 in a validator's account was not earning rewards. With EIP-7251, validators can bond and earn on a much larger amount, up to 2048 ETH (Ethereum Pectra Upgrade: Key Improvements and Impact | QuickNode Guides). In effect, it allows *consolidation* of stakes: one validator can now represent what used to require many separate 32 ETH validator slots.

Relevance to Pectra: This is one of the headline changes in the Pectra upgrade for Ethereum's consensus layer. By lifting the limit, Pectra enables two big outcomes: (1) **Large stakers can run fewer validators** – reducing overhead on the network – and (2) **Small stakers benefit from compounding** – now if they add a bit more ETH or let rewards accumulate above 32, that extra starts earning rewards too (Ethereum Pectra Upgrade: Key Improvements and Impact | QuickNode Guides) (Ethereum Pectra Upgrade: What it means for stakers). From a network perspective, fewer total validators means fewer signatures and messages each epoch, easing network traffic and validator duty load. Pectra is implementing this to address the scalability issue of having millions of validators; it's a proactive step to avoid hitting technical limits on consensus communications (Upcoming Ethereum Upgrades & Catalysts | Galaxy).

Analogy: Imagine a nightclub where each table could only seat 32 people, so if a group of 100 came in, they'd split into 4 tables and a bit of overflow. This leads to lots of tables to serve. If the club expands table size to seat up to 2048, that same group can sit at one table. Service (consensus) becomes simpler with fewer tables, and groups don't have to split up. In Ethereum terms, a big operator like a staking pool or exchange could merge many 32 ETH validators into one validator (up to 2048 ETH), which is much less strain to manage. Similarly, an individual who started with 32 ETH can top up their validator with more ETH over time instead of creating a new validator every 32 ETH.

Historical Context & Problem It Solves: The 32 ETH limit was set in 2020 for the Beacon Chain launch to encourage decentralization – the idea was to have many modest-sized validators rather than a few huge ones (Upcoming Ethereum Upgrades & Catalysts | Galaxy). This worked (Ethereum now has over half a million active validators), but it also led to concerns about network overhead and inefficiencies as the validator count soared past expectations (Upcoming Ethereum Upgrades & Catalysts | Galaxy). Studies indicated that the networking load could become unsustainable beyond ~1.4 million validators (Upcoming Ethereum Upgrades & Catalysts | Galaxy). EIP-7251 addresses this by nudging the ecosystem toward **consolidation**: large stakers will consolidate stakes (since they can earn more efficiently by maxing out each validator), thereby slowing the growth of validator count. It also introduces the concept of *compounding* rewards on a single validator – validators can now automatically grow their effective balance as they earn, up to the cap. This EIP is somewhat controversial in terms of decentralization (fewer, bigger validators), but it is generally seen as necessary maintenance for Ethereum's health, and Pectra implements it to ensure the network can continue to scale without breaking the consensus layer.

EIP-7685: General-Purpose Execution Layer Requests

Definition: EIP-7685 establishes a standardized framework for the Execution Layer (EL) to send requests to the Consensus Layer (CL) as part of each block. It does this by adding a new field

(often called `requests_hash`) to the block header which represents a list of *requests*. Each request has a type code and associated data. This framework is *general-purpose*, meaning it's designed to handle many kinds of cross-layer operations (deposits, validator exits, etc.) in a unified way (EIP-7685) (EIP-7685).

Relevance to Pectra: Pectra uses EIP-7685 as the backbone for several other improvements. Both the on-chain deposits (EIP-6110) and triggerable exits (EIP-7002) rely on this new request system to function. By including 7685, Pectra essentially **bridges the gap** between Ethereum's two layers, allowing the execution side (smart contracts and user transactions) to directly invoke consensus-layer actions. For example, a smart contract can create a "withdrawal request" that the Beacon Chain will process, or include deposit info without a separate voting mechanism (Ethereum Pectra Upgrade: Key Improvements and Impact | QuickNode Guides). This makes cross-layer features consistent and easier to implement, enabling future upgrades (like maybe validator *consolidation operations or other admin tasks) to reuse the same pipeline. In short, EIP-7685 is included in Pectra to streamline how Ethereum handles any operation that spans the EL and CL, improving efficiency and opening the door to new functionality.

Analogy: Consider a large organization that until now had no formal channel for communication between two departments – employees were using ad-hoc methods (emails, manual forms, personal contacts) for each specific task. EIP-7685 is like setting up a **standard ticketing system** between the departments. Now if one side needs something from the other (whether it's a new hire request, a data report, or equipment purchase), they all go through the same portal with a request type. This reduces confusion and errors. Similarly, Ethereum had different mechanisms for each EL↔CL interaction (deposits were one-off, exits another, etc.), which was inefficient. Now with a unified request system, it's much clearer and more robust – any new cross-layer feature can just plug into this pipeline.

Historical Context & Problem It Solves: When Ethereum's Beacon Chain (CL) was separate from the old ETH1 (EL), interactions were bolted on in a minimal way (like the deposit contract) because the two systems were distinct. After the Merge unified them, it became apparent that a better integration was possible. EIP-7685 emerged to tackle the *lack of a standardized EL→CL communication method* (Ethereum Pectra Upgrade: Key Improvements and Impact | QuickNode Guides). Without it, each new feature (such as contract-initiated validator actions) would require a bespoke solution. This EIP is co-authored by core devs involved in the Merge and staking, indicating it was part of resolving technical debt and enabling future governance automation. The result is a more flexible Ethereum: Pectra's adoption of 7685 removes prior inefficiencies and allows smart contracts to safely interact with consensus-layer operations in a controlled manner (Ethereum Pectra Upgrade: Key Improvements and Impact | QuickNode Guides), something not possible before. It essentially future-proofs the protocol for more complex validator management and cross-layer features down the line.

EIP-7702: Temporary Smart Accounts (EOA with Code)

Definition: EIP-7702, often described as an *account abstraction* proposal, lets a regular Externally Owned Account (EOA – a normal user wallet) temporarily execute custom smart contract code during a transaction. It introduces a new transaction type where the sender's account can have a piece of code attached for that one transaction, effectively behaving like a smart contract account for the scope of that transaction (Ethereum Pectra Upgrade: Key Improvements and Impact | QuickNode Guides). After the transaction, the account goes

back to being a normal EOA with no code. This does **not** convert EOAs into permanent contracts, but gives them a one-time "smart" capability when needed.

Relevance to Pectra: This is one of the most anticipated features in the Pectra upgrade for everyday users. By including EIP-7702, Pectra significantly improves **wallet flexibility and user experience**. Wallets can leverage this to enable things like: **batched transactions** (e.g. perform multiple actions in one go), **gas fee sponsorship or payment in ERC-20 tokens** (someone else or another token pays the fee), and **advanced security logic** (like requiring multiple signatures or constraints, without a separate contract wallet) (Ethereum Pectra Upgrade: Key Improvements and Impact | QuickNode Guides) (Ethereum Pectra Upgrade | Coinbase). In essence, Pectra's account abstraction measure bridges the gap between simple accounts and fully programmable contract accounts (Ethereum Pectra Upgrade: Key Improvements and Impact | QuickNode Guides). Users won't notice the technical details, but they'll enjoy more seamless interactions – imagine confirming one action in your wallet that actually does a series of complex operations on-chain, made possible because your EOA can temporarily run a tiny program that you specify.

Analogy: It's like giving your ordinary bank account a one-time upgrade to function like a business account with special powers for a specific task. Suppose normally you could only send one payment at a time (EOA behavior), but now you can attach a short script to your account that says "split this payment, swap some currency, then send to multiple recipients, and have someone else cover the fees." For that single transaction, your account behaves like a full-fledged automated agent (much like a smart contract). After it's done, your account is back to normal. Another analogy: EOAs are like simple phones with no apps, and EIP-7702 allows you to install a mini-app just for the duration of a call – suddenly you can do smartphone-like things temporarily. This makes previously complex interactions much more user-friendly.

Historical Context & Problem It Solves: Ethereum developers have long sought *account abstraction*, which means making user accounts more powerful and flexible (so we're not limited to the rigid "one signature = one transaction" model inherited from Bitcoin). Earlier attempts included proposals like EIP-293 (which didn't materialize) and the later **EIP-4337**, which implemented smart accounts via an off-chain infrastructure of "bundlers" and alternate mempools without changing the protocol. EIP-7702 is a direct protocol change to realize account abstraction goals (Ethereum Pectra Upgrade: Key Improvements and Impact | QuickNode Guides). The problem it addresses is the cumbersome user experience of Ethereum: for example, needing to send multiple transactions for a multi-step operation, or not being able to have others pay your gas easily. Before Pectra, achieving those features required deploying a custom smart contract wallet (with schemes like Argent or Gnosis Safe) or using 4337's user operations – both complex solutions. With EIP-7702, Ethereum natively allows a user's account to *act like a smart contract when needed*, making advanced wallet features simpler to implement and more accessible. It's a step towards reducing the distinction between user accounts and contract accounts, which over time can greatly simplify how users interact with dApps (less signing confusion, more safety checks in a single transaction, etc.). In short, Pectra's inclusion of 7702 is a big UX win, bringing Ethereum closer to mainstream-friendly interactions by empowering wallets with new superpowers (Ethereum Pectra Upgrade: Key Improvements and Impact | QuickNode Guides) (Ethereum Pectra Upgrade | Coinbase).

EIP-7742: Dynamic Blob Capacity Adjustment

Definition: EIP-7742 enables the Ethereum network to **dynamically adjust the maximum and target number of blob transactions per block** via the Consensus Layer, rather than having these limits fixed in the code. “Blobs” refer to large data chunks introduced in proto-danksharding (EIP-4844, part of the prior Dencun upgrade) that layer-2 rollups use for cheap data storage. Previously, the blob capacity was hard-coded (e.g., target of 3 blobs and max of 6 blobs per block). With EIP-7742, the Beacon Chain can modify these parameters over time without requiring a hard fork on the Execution Layer (Upcoming Ethereum Upgrades & Catalysts | Galaxy). In other words, it decouples the blob limit settings between the EL and CL.

Relevance to Pectra: This EIP is included in Pectra to **future-proof Ethereum’s data scalability**. Right away, it lays the groundwork for increasing data throughput on-chain. In fact, developers have signaled plans to raise the blob throughput (for example, possibly targeting 5 blobs on average, 8 max, or even 6/9) using this mechanism soon (Upcoming Ethereum Upgrades & Catalysts | Galaxy) (Upcoming Ethereum Upgrades & Catalysts | Galaxy). By making blob capacity tunable, Ethereum can respond to layer-2 demand and scale up data availability for rollups without another full upgrade. Pectra’s focus is partly on preparing for bigger changes (like full danksharding later), and EIP-7742 is a key part of that “prepare for future upgrades” objective. It directly **improves data availability** flexibility, which benefits rollups (they’ll have more room for posting transaction data as needed, and fees can be better managed by adjusting targets). As a result, users should see cheaper and more plentiful layer-2 capacity over time, keeping Ethereum L2s competitive and growth-ready.

Analogy: Think of a highway where the speed limit and lane count were previously fixed by law – changing them meant a long legislative process. EIP-7742 hands that control to a dynamic system, like digital road signs that can raise the speed limit or open an extra lane during rush hour. This means Ethereum can **adjust its bandwidth on the fly** to meet traffic needs. For example, if the network sees sustained high demand from rollups, the consensus layer could decide to allow more blobs per block (akin to opening more lanes), increasing throughput without everyone needing to upgrade their software at that moment. It’s a flexible mechanism similar to how EIP-1559 adjusts gas block sizes with demand, but here applied to these new blob data containers.

Historical Context & Problem It Solves: The introduction of blob-carrying transactions (EIP-4844) was a massive leap for Ethereum’s scaling, but it came with conservative defaults (to ensure stability). Prior to EIP-7742, if Ethereum wanted to increase or tweak blob limits, it needed a coordinated hard fork of both execution and consensus layers because both had those numbers baked in. This was not ideal for agile improvements. EIP-7742 decouples those settings so that the consensus layer can dictate blob limits going forward (Upcoming Ethereum Upgrades & Catalysts | Galaxy). This means small increases (or future algorithmic adjustments) to data capacity don’t have to wait for full client releases on the EL. It addresses a governance and technical bottleneck, making the network more adaptable. In the context of Pectra, it “lays the foundation for planned blob changes in the future” (Ethereum Pectra Upgrade) – essentially ensuring that as demand for data inclusion grows (with more rollup usage), Ethereum can smoothly scale up without the risk of fragmentation between EL and CL. This is a strategic upgrade that keeps Ethereum’s rollup-centric roadmap on track, preventing the data availability limit from becoming the next scalability chokepoint.

Security Issues Identified in Pectra's EIPs

EIP-2935 (Historical Block Hashes) received audits from Dedaub and Sigma Prime. Dedaub identified an advisory-level issue where the ring buffer implementation stored 8,191 blocks instead of the 8,192 defined in the EIP. While this was a minor deviation, it didn't introduce any security risks. Sigma Prime highlighted two concerns: first, an inconsistency between the specification and implementation (PEC-04), which was addressed through clarification; and second, a potential overflow when `block.number == 0` (PEC-05), which could cause `block.number - 1` to underflow. Although considered low impact, this remains unresolved and flagged for edge-case hygiene.

EIP-7002 (Execution-Layer-Triggered Withdrawals) had the most notable findings. Dedaub raised a medium-severity issue involving denial-of-service risk—only 16 validator withdrawal requests can be processed per block, which makes the queue susceptible to spam attacks that could delay legitimate exits. They also flagged two advisory-level issues around gas inefficiencies and lack of clarity in the exponential fee mechanism used to mitigate spam. Sigma Prime reinforced the DoS concern with PEC-01, showing that attackers could batch hundreds or thousands of withdrawal requests into a single block without paying proportionate gas fees. This exploit risks bloating storage and undermining network fairness. PEC-02 identified that the withdrawal contract unexpectedly accepts ETH, which could lead to confusion or even trapped funds. PEC-03 described a theoretical overflow risk in the fee calculation function, which could be exploited in extreme cases, although this was classified as informational due to its impracticality under normal conditions.

EIP-7251 (Validator Balance Increase), which allows validators to stake up to 2048 ETH, inherits a similar threat model. Dedaub highlighted that the validator consolidation queue can also be spammed, creating a bottleneck for legitimate consolidations. This was given a medium severity rating. Additional advisory comments mentioned unnecessary storage usage and redundant state writes that could be optimized to save gas. Sigma Prime reiterated the concern in PEC-01 (also applied to EIP-7002), warning that validator consolidation operations could be batched in a way that breaks intended fee incentives and delays honest users from accessing the mechanism.

EIP	About	Why it Matters
EIP-7702: Set EOA account code - Biggest wallet and UX improvement	This EIP introduces a new trasaction type that will allow externally owned accounts (EOAs) to temporarily act as smart contracts for the duration of a transaction. This will be achieved by enabling EOAs to set and execute smart contract code without permanently converting into a contract account. The change will enable wallets (EOAs) to execute batch transactions, enable sponsored gas payments (meta-transactions), and allow for implementation of custom validation logic.	This change will improve user experience, wallet functionality, and gas efficiency by reducing the number of steps required for end-users to interact with DeFi protocols, onchain games, and other dapps. Dapp users will no longer need to leave an application to complete transactions–reducing the likelihood of user drop-off and creating a more seamless onchain experience. The change will bring Ethereum closer to full account abstraction, making the network more flexible, efficient, and user-friendly.
EIP-7251: Increase the MAX_EFFECTIVE _BALANCE - Biggest staking change	Primarily impacts: staking efficiency, validator operations and management, staking liquidity and exits This enables raising the validator maximum effective balance ('MaxEB') from 32 ETH to 2,048 ETH. To opt in, validators will need to update their withdrawal credentials to 0x02. The MaxEB change will allow validators to earn rewards on any balance between 32 and 2,048 ETH, giving larger stakers (e.g., institutional investors) and staking providers the ability to consolidate their validators. For 0x02 validators, the auto-sweep of balances (rewards) >32 ETH will change to activating when balances exceed 2,048 ETH, enabling rewards compounding. To mitigate the heightened risk associated with large balances, the initial slashing penalty for validators employing the new MaxEB parameter will be 1/4,096 of their effective balance, in lieu of the current 1/32. The new constant MIN_ACTIVATION_BALANCE will also be introduced to set a formal minimum activation balance of 32 ETH. EIP-7251 will also change exit and withdrawal mechanics to prevent large fluctuations in staked ETH. The exit queue will no longer be governed by a churn limit denominated in number of validators per epoch (currently 16). Instead, the total churn per epoch will be based on the amount of ETH to leave the network. Churn will be hard capped at 256 ETH per epoch (~57,600 ETH per day).	The change to MaxEB will enable rewards compounding, as stakers will be able to reinvest rewards into the validator balance until it reaches 2,048 ETH. The ability to consolidate validators will simplify operations and reporting and improve flexibility for stakers and staking operators. Consolidation will also reduce the total number of validators in the network (at least temporarily), helping lower the attestation load and reduce network overhead. This will improve consensus efficiency and staking scalability.
EIP-7002: Execution layer triggerable withdrawals - Biggest flexibility change for staking	Primarily impacts: validator withdrawals and exits, staking security, control over funds Under the current configuration, validator exits and withdrawals can only be triggered from the consensus layer (CL), using a validator's active key. This EIP will add a new mechanism that will allow validators to manually trigger withdrawals and exits from the execution layer (EL) using their EL (0x01) withdrawal credentials (e.g., via an EOA or smart contract). Under the new system, exit request transactions will be submitted to the EL then processed by	Limiting withdrawal and exit triggering to only a validator's active key can pose limitations and unnecessary risk. Because active keys sign and perform validator duties, they are hot and can have security vulnerabilities. Withdrawal credentials, on the other hand, can remain cold when used only for withdrawal and ownership operations. Enabling the use of withdrawal credentials for exits and withdrawals will remove absolute reliance on

	the CL (validators will join the exit queue, just like in a CL triggered exit). For withdrawals, requests will be submitted to the EL then included in the EL manual withdrawal queue (which will allow 8 withdrawals per block). This new mechanism will be in addition to, not in lieu of, CL triggered withdrawals and exits.	potentially vulnerable active keys. Eliminating this reliance will also reduce trust dependencies for stakers who delegate to providers and will lessen the impact in the event of an active key loss. The change will make Ethereum's staking model more secure, flexible, and trustless overall.
EIP-7691: Blob throughput increase - Biggest impact on L2 scalability	Primarily impacts: rollup performance, network scalability This EIP will increase blob throughput by raising the target number of blobs per block from 3 to 6 and the maximum from 6 to 9. This proposal builds on EIP-4844 from the Dencun upgrade, which introduced blob transactions. Blob transactions allow rollups to store large amounts of data more efficiently in temporary "blobs" (rather than calldata), significantly reducing gas fees and network congestion. EIP-7691 will allow blobs to accommodate more data in each block, improving data availability for rollups. The blob base fee update mechanism will also be adjusted to ensure that fee changes remain responsive to network usage under the new blob limits. **(i)** Full blobs: Base fee increases by ~8.2% (currently ~12.5%) **(ii)** No blobs: Base fee decreases by ~14.5% (currently ~11.1%) Note that this change may also impact validators and node operators, as they will need to handle larger blocks with more blobs–potentially increasing bandwidth and storage requirements.	Increasing blob throughout will allow rollups to process more transactions efficiently, potentially leading to reduced congestion and improved transaction speeds for users of those networks. With increased data availability, rollups will be able to operate more efficiently, potentially leading to lower gas fees. This change will enhance Ethereum's capacity to support L2s, thereby improving its scalability. Note that this change is a short-term solution to enhance scalability until more comprehensive solutions are deployed.
EIP-6110: Supply validator deposits on chain	Primarily impacts: validator activation and deposit time, deposit security This will integrate validator deposit functionality directly into Ethereum's Execution Layer (EL) blocks and eliminate the current Consensus Layer (CL) proposer voting mechanism.	Elimination of Eth1Data polling will significantly decrease deposit processing time–taking it from ~12 hours down to ~13 minutes (assuming no validator queue). Removing reliance on proposer voting will also significantly increase deposit security by eliminating any risk of adversarial behavior. This change will also reduce the engineering and design complexity of CL client software, improving network resilience and efficiency.
EIP-7549: Move committee index outside Attestation	Primarily impacts: validator performance, network overhead Ethereum validators are grouped into committees to submit attestations. When a validator signs an attestation, the index field (representing the committee index) is included to indicate which committee the validator was assigned to for the given epoch. Currently, the committee index field resides inside the signed Attestation message. This means that two validators that vote identically but belong to different committees will have different signatures because their index values are included in their signatures. This upgrade will relocate the index field to outside of the signed Attestation message, making it so that identical votes across different committees produce the same signature.	This change will make attestation aggregation and verification more efficient. It will reduce bandwidth and processing overhead for validators, helping them operate more efficiently. Stakers won't see direct changes (no changes to validator behavior, rewards, or slashing), but validators will benefit from a more optimized consensus process. This should reduce network load and enhance flexibility for staking providers while keeping Ethereum's consensus secure and stable.