LP-300: The Lux Schema Registry

One source of truth for TypeKind, ShapeKind, and EVM precompile slot allocations

Zach Kelling

Lux Industries

2026-06-03

Abstract

The LP corpus allocates ZAP schema IDs, TypeKind bytes, ShapeKind bytes, and EVM precompile slots in fragments scattered across 25 LPs. LP-186 owns `0xC0..0xCB` for chains-VMs. LP-201 owns `0xD0..0xDF` for P2P stream types. LP-208 originally claimed `0xF0..0xFF` for DAG mempool, then reclaimed `0xE0..0xE1` after LP-218 rollup envelopes shipped. LP-211 claims `0xE2..0xE7` for cross-shard 2PC. LP-218 Draft initially claimed `0xE0..0xEF` for rollup envelopes – which overlapped LP-208 *and* LP-211. LP-214 light client claims `0xF0..0xF2`. LP-219 reuses LP-211's `0xE2..0xE7` at the rollup tier. LP-0120 owns the `0x012201..0x012208` PQ precompile range, and LP-218 / LP-220 attach P3Q wrappers on three of those slots. LP-078 owns the broad EVM precompile suite spanning classical curves, DEX, AI, hashing, FHE, ZK, threshold sigs, and KZG. This paper collapses every allocation into one master registry partitioned by three orthogonal namespaces (TypeKind, ShapeKind, Precompile slot), resolves four pre-existing conflicts, and pins the cross-referencing pattern that every consuming LP MUST follow. LP-300 is normative; consuming LPs are informative. Final from genesis (unix `1766708400` = 2025-12-25T16:20:00-08:00): the registry ships no wire change, only authoritative documentation, so no migration window is required.

Contents

1 Introduction

LP-300 is a meta-registry. It allocates no new wire bytes, no new precompile slots, no new schemas. Its sole job is to record – in one normative place – every allocation made by every other LP in the corpus. Where a consuming LP says "this draft uses TypeKind `0xE8`", LP-300 says "`0xE8` is owned by LP-218; here is the row."

1.1 Why a master registry

Three years of LP authoring have accumulated 25 LPs that touch one of three address spaces – the ZAP TypeKind byte, the per-TypeKind ShapeKind byte, or an EVM precompile slot. The allocations live in the "Wire schemas" section, the "Precompile slot" section, the "Schema-ID allocation map", or scattered tables in the LP body. Three concrete failure modes follow:

1. **Allocation drift.** LP-208 originally allocated `0xF0..0xFF` for DAG mempool. LP-214 then allocated `0xF0..0xF2` for light client. The conflict went undetected until LP-218 also targeted the same range. LP-208 quietly re-homed to `0xE0..0xE1` – but the original Draft text in several places still cites `0xF0`.
2. **Slot triple-claim.** LP-208 (`0xE0..0xE1`), LP-211 (`0xE2..0xE7`), and an earlier LP-218 Draft (`0xE0..0xEF`) all claimed bytes inside `0xE0..0xEF`. Three Drafts cannot all be Final without arbitration.
3. **Wrapper-over-primitive ambiguity.** LP-0120 owns the PQ precompile slot `0x012205` as Plonky3-fork STARK. LP-218 attaches a P3Q wrapper on the same slot. Two LPs nominally "own" the same hex digit. The wrapper-vs-primitive distinction is real but invisible without a registry that records both.

The fix is decomplecting: separate *authoritative allocation* (LP-300) from *wire behavior specification* (the consuming LP). A consuming LP becomes informative-only on the allocation question. LP-300 wins any disagreement.

1.2 Three orthogonal namespaces

LP-300 partitions every allocation into three orthogonal namespaces plus one out-of-scope namespace. Cross-namespace collisions are impossible by construction because each lives in a different address space:

#	Namespace	Width	Carrier
1	TypeKind byte	1 byte (`0x00-0xFF`)	byte 0 of every ZAP envelope on the wire
2	ShapeKind byte	1 byte, namespaced under each TypeKind	byte 1 of every ZAP envelope (per TypeKind table)
3	Precompile slot	uint16 / uint24 / uint160	EVM precompile address
4	Genesis assetID / chainID (out of scope)	uint32 / uint160	per-chain genesis blob

Table 1: The four allocation namespaces. LP-300 is normative for namespaces 1, 2, and 3; namespace 4 is owned by LP-019 / LP-099 / LP-134 and is referenced here only to disclaim overlap. The terms "schema ID" and "TypeKind byte" are used interchangeably throughout the LP corpus; they name the same byte.

1.3 What this paper is not

LP-300 is a reference document. It contains no measured benches: the registry ships no wire change, so there is nothing to measure. It contains no engineering tradeoffs: the tradeoffs were made by the consuming LPs and are documented there. It contains no operator-facing config: nothing in LP-300 is configurable at runtime.

What LP-300 *does* contain: every byte allocated, every slot allocated, the owner LP for each row, the conflict resolution log for the four pre-existing conflicts, the cooldown policy for deprecated allocations, and the cross-referencing pattern every consuming LP MUST follow.

1.4 Final from genesis

LP-300 activates at the genesis of the new final Lux network – unix `1766708400` = 2025-12-25T16:20:00-08:00 – with no migration window. A meta-registry that ships no wire change has nothing to migrate; it is Final the moment the network is. Any post-activation amendment is an additive new row, never a re-allocation of a live slot. Re-allocation of a deprecated slot requires the one-year cooldown documented in §8.

2 Namespace 1 – TypeKind byte

The TypeKind byte (also called the "schema ID") is byte 0 of every ZAP envelope on the wire. It identifies the top-level message type and selects the per-TypeKind ShapeKind table (§3). LP-022 defines the framing; LP-200 layers the discriminator on top; LP-300 owns the allocations.

The 256-byte space is partitioned into six blocks by allocator-LP. Reserved ranges in between are held for future expansion and MUST NOT be allocated except by an LP-300 amendment.

2.1 Consensus-wire block (LP-182, `0x01-0x0F`)

LP-182 owns the 14-row consensus-wire block. These envelopes carry QuasarCerts, blocks, witness proofs, and the per-leg signatures that compose the cert.

Hex	Owner LP	Purpose	Status	Verified
`0x00`	reserved	Null / illegal envelope	Reserved	2026-06-03
`0x01`	LP-182	`QuasarCert`	Final	2026-06-03
`0x02`	LP-182	`QBlock`	Final	2026-06-03
`0x03`	LP-182	`WitnessProof`	Final	2026-06-03
`0x04`	LP-182	`MagnetarAggregateCert`	Final	2026-06-03
`0x05`	LP-182	`PolarisLegs`	Final	2026-06-03
`0x06`	LP-182	`QuasarSig`	Final	2026-06-03
`0x07`	LP-182	`EpochBundle`	Final	2026-06-03
`0x08`	LP-182	`PrismCut`	Final	2026-06-03

Hex	Owner LP	Purpose	Status	Verified
0x09	LP-182	StakeWeightedCut	Final	2026-06-03
0x0A	LP-182	TxAuthEnvelope	Final	2026-06-03
0x0B	LP-182	PQPermit	Final	2026-06-03
0x0C	LP-182	DAGVertex	Final	2026-06-03
0x0D	LP-182	PolicyCert	Final	2026-06-03
0x0E-0x0F	LP-182	Reserved for consensus-wire growth	Reserved	2026-06-03

Table 2: Consensus-wire TypeKinds owned by LP-182. The 0x0E and 0x0F reserves are LP-182's growth budget; an LP-300 amendment is required to fill them.

2.2 Broad protocol reserve (0x10-0xBF)

The largest contiguous reserve in the TypeKind space – 176 bytes – held for future protocol-level allocations. No LP owns this range today. Likely landing zones, none committed, are future chain-internal schemas analogous to LP-182, future P2P transport primitives analogous to LP-201, and future signed-tx envelope variants. Any consuming LP that wants to land here MUST open an LP-300 amendment first.

2.3 Chains-VM block (LP-186, 0xC0-0xCF)

LP-186 owns the 12 chains-VM wire types plus 4 reserves. Each row delegates the per-VM ShapeKind table to the owning chains-VM specification.

Hex	Owner LP	Purpose	Status	Verified
0xC0	LP-186	aivm chains-VM wire	Draft	2026-06-03
0xC1	LP-186	bridgevm chains-VM wire	Draft	2026-06-03
0xC2	LP-186	dexvm chains-VM wire	Draft	2026-06-03
0xC3	LP-186	evm chains-VM wire (chains-side glue)	Draft	2026-06-03
0xC4	LP-186	graphvm chains-VM wire	Draft	2026-06-03
0xC5	LP-186	identityvm chains-VM wire	Draft	2026-06-03
0xC6	LP-186	keyvm chains-VM wire	Draft	2026-06-03
0xC7	LP-186	oraclevm chains-VM wire	Draft	2026-06-03

Hex	Owner LP	Purpose	Status	Verified
0xC8	LP-186	quantumvm chains-VM wire	Draft	2026-06-03
0xC9	LP-186	relayvm chains-VM wire	Draft	2026-06-03
0xCA	LP-186	thresholdvm chains-VM wire	Draft	2026-06-03
0xCB	LP-186	zkvm chains-VM wire	Draft	2026-06-03
0xCC-0xCF	LP-186	Reserved for chains-VM growth	Reserved	2026-06-03

Table 3: Chains-VM TypeKinds owned by LP-186. ShapeKind tables under each 0xCx TypeKind are delegated to the owning chains-VM and documented in LP-186 and the per-VM wire files; they are not re-listed in LP-300.

2.4 P2P stream-type block (LP-201, 0xD0-0xDF)

LP-201 owns the 16-row P2P stream-type block. Each P2P TypeKind is degenerate at the ShapeKind level – one TypeKind, one shape – so no separate ShapeKind table is needed for this block.

Hex	Owner LP	Purpose	Status	Verified
0xD0	LP-201	ConsensusVote	Draft	2026-06-03
0xD1	LP-201	ConsensusProposal	Draft	2026-06-03
0xD2	LP-201	ConsensusCert (mirrors 0x01 QuasarCert)	Draft	2026-06-03
0xD3	LP-201	BlockRequest	Draft	2026-06-03
0xD4	LP-201	BlockResponse	Draft	2026-06-03
0xD5	LP-201	TxGossip	Draft	2026-06-03
0xD6	LP-201	TxRequest	Draft	2026-06-03
0xD7	LP-201	TxResponse	Draft	2026-06-03
0xD8	LP-201	FilterAdvertise (P2P bloom-filter gossip)	Draft	2026-06-03
0xD9	LP-201	SnapshotAdvertise (P2P state-sync gossip)	Draft	2026-06-03
0xDA	LP-201	ChunkRequest (content-addressed fetch)	Draft	2026-06-03
0xDB	LP-201	ChunkResponse (content-addressed fetch)	Draft	2026-06-03

Hex	Owner LP	Purpose	Status	Verified
0xDC	LP-201	PeerInfo (signed peer record)	Draft	2026-06-03
0xDD	LP-201	DHTFindNode (Kademlia)	Draft	2026-06-03
0xDE	LP-201	DHTFindValue (Kademlia)	Draft	2026-06-03
0xDF	LP-201	DHTStore (Kademlia)	Draft	2026-06-03

Table 4: P2P stream-type TypeKinds owned by LP-201. Each row carries exactly one shape; no per-TypeKind ShapeKind table.

2.5 Mempool / 2PC / rollup block (0xE0-0xEF)

The most-contested block in the corpus. Three LPs originally claimed overlapping bytes here; LP-300 resolves the contention per the LP-214 canonical handoff (§5). The final allocation: LP-208 takes 0xE0..0xE1, LP-211 takes 0xE2..0xE7, LP-218 takes 0xE8..0xEF (shifted from prior 0xE0..0xE3 claim). LP-219 reuses LP-211's 0xE2..0xE7 at the rollup tier as a ShapeKind-level extension (TxKind 0x44), not a TypeKind-level reclamation.

Hex	Owner LP	Purpose	Status	Verified
0xE0	LP-208	DAGHeader (mempool) – reclaimed from prior 0xF0	Draft	2026-06-03
0xE1	LP-208	DAGBody (mempool) – reclaimed from prior 0xF1	Draft	2026-06-03
0xE2	LP-211	CrossShardTxGroup – reused by LP-219 at rollup tier	Draft	2026-06-03
0xE3	LP-211	CrossShardDispatch – reused by LP-219 at rollup tier	Draft	2026-06-03
0xE4	LP-211	CrossShardPrepareAck – reused by LP-219 at rollup tier	Draft	2026-06-03
0xE5	LP-211	CrossShardRefuseAck – reused by LP-219 at rollup tier	Draft	2026-06-03
0xE6	LP-211	CrossShardCommitNotify – reused by LP-219 at rollup tier	Draft	2026-06-03
0xE7	LP-211	CrossShardAbortNotify – reused by LP-219 at rollup tier	Draft	2026-06-03
0xE8	LP-218	RollupBatchTx – shifted from prior 0xE0	Draft	2026-06-03
0xE9	LP-218	RollupChallengeTx – shifted from prior 0xE1	Draft	2026-06-03
0xEA	LP-218	RollupExitTx – shifted from prior 0xE2	Draft	2026-06-03
0xEB	LP-218	RollupGenesisTx – shifted from prior 0xE3	Draft	2026-06-03

Hex	Owner LP	Purpose	Status	Verified
0xEC	LP-218 / LP-219	RollupAggregateBatch – LP-219 TxKind 0x44 lives here	Draft	2026-06-03
0xED-0xEF	LP-218	Reserved (sequencer rotation, cross-rollup commit)	Reserved	2026-06-03

Table 5: Post-resolution 0xE0-0xEF block. LP-208 holds 0xE0-0xE1; LP-211 holds 0xE2-0xE7; LP-218 holds 0xE8-0xEF. LP-219 is a ShapeKind-level extension on top of LP-211 envelopes at the rollup tier; see §3.

2.6 Light client block (LP-214, 0xF0-0xF2)

LP-214 owns the three light-client TypeKinds with paired ShapeKinds in the 0x50-0x52 band.

Hex	Owner LP	Purpose	Status	Verified
0xF0	LP-214	LightClientBlockHeaderRequest	Draft	2026-06-03
0xF1	LP-214	LightClientStateProofRequest	Draft	2026-06-03
0xF2	LP-214	LightClientCertProof	Draft	2026-06-03
0xF3-0xFF	reserved	Reserved for future framing – DO NOT ALLOCATE	Reserved	2026-06-03

Table 6: Light-client TypeKinds owned by LP-214 plus the top-of-namespace reserve. 0xF3-0xFF is the final 13-byte reserve in the TypeKind space.

3 Namespace 2 – ShapeKind byte

The ShapeKind byte is the discriminator at byte 1 of every ZAP envelope, namespaced under each TypeKind. The ShapeKind value space is fresh per TypeKind – the same byte value can mean different things under different TypeKinds without collision.

LP-300 records only the *cross-LP-visible* ShapeKinds – those that pass between LPs at the consensus/wire boundary. Per-VM TxKind tables (e.g., dexvm OrderTx / CancelTx / SwapTx) are internal to the owning chains-VM and are documented in LP-186 plus the per-VM wire files; they are NOT re-listed in LP-300.

3.1 Cross-LP-visible ShapeKinds

3.2 Delegated ShapeKind tables

Two TypeKind blocks delegate their ShapeKind tables out of LP-300 entirely:

- **0xC0-0xCB (chains-VM, LP-186).** Each chains-VM TypeKind has its own ShapeKind table for the VM's internal tx types. These are documented in LP-186 and the per-VM wire files. Cross-VM TxKind collisions are impossible because each VM's ShapeKind space is local to its TypeKind.

TypeKind	ShapeKind	Owner LP	Purpose	Status
0xE8	0x40	LP-218	RollupBatchTx body kind	Draft
0xE9	0x41	LP-218	RollupChallengeTx body kind	Draft
0xEA	0x42	LP-218	RollupExitTx body kind	Draft
0xEB	0x43	LP-218	RollupGenesisTx body kind	Draft
0xEC	0x44	LP-219	RollupAggregateBatch body kind (cross-rollup atomic)	Draft
0xF0	0x50	LP-214	LightClientBlockHeaderRequest body kind	Draft
0xF1	0x51	LP-214	LightClientStateProofRequest body kind	Draft
0xF2	0x52	LP-214	LightClientCertProof body kind	Draft

Table 7: The eight cross-LP-visible ShapeKinds. LP-218 owns the 0x40-0x43 band under TypeKinds 0xE8-0xEB. LP-219 adds 0x44 under TypeKind 0xEC as the cross-rollup atomic extension that reuses LP-211 semantics at the rollup tier. LP-214 owns the 0x50-0x52 band under TypeKinds 0xF0-0xF2.

- **0xD0-0xDF (P2P, LP-201).** Each P2P TypeKind is degenerate at the ShapeKind level – one TypeKind, one shape. No per-TypeKind ShapeKind table exists or is needed. The ShapeKind byte position is always 0x00 for these envelopes.

LP-300's job for these blocks is to record *that* the delegation exists, not the contents of the delegated tables.

4 Namespace 3 – EVM Precompile slot

EVM precompiles live in three disjoint address bands and a handful of legacy / stateful clusters. Each row is owned by exactly one LP ("Owner LP") and optionally co-owned by a wrapper LP that attaches a higher-level construction on top of the raw primitive.

The address bands and their authoritative owners:

- 0x000B..0x0FFF – Ethereum-compatible / Lux core (LP-078)
- 0x011xxx – Federation / governance (LP-0011)
- 0x012xxx – Post-quantum cryptography (LP-0120)
- 0x02xxxxxx – Subnet-EVM-style stateful range (LP-3520 / LP-078)
- 0x6010..0x9FFF – Application-layer (DEX, AI)
- 0xB000..0xBFFF – Blob / KZG
- 0x020000+ – Reserved for future precompile blocks

4.1 Classical EVM core + Ethereum-compat (LP-078)

The broadest band. LP-078 is the authoritative owner of every slot listed here; per-row Owner LP citations name the primitive's specifying LP where one exists.

Slot	Owner LP	Name	Purpose	Status
0x000B	LP-078	BLS12-381 G1 add	EIP-2537	Final
0x000C	LP-078	BLS12-381 G1 scalar mul	EIP-2537	Final
0x000D	LP-078	BLS12-381 G1 MSM	EIP-2537	Final
0x0100	LP-078	secp256r1 / P-256	EIP-7212 (RIP-7212)	Final
0x0300	LP-078	AI mining	LP-009 family	Draft

Slot	Owner LP	Name	Purpose	Status
0x0300..01	LP-078 / LP-127	TEE attestation (NVTrust, SGX, SEV-SNP, TDX)	LP-127	Draft
0x0300..10	LP-078	AI compute marketplace	LP-130	Draft
0x0300..20	LP-078	Quasar Verkle tree verify	LP-026	Draft
0x0300..21	LP-078	Quasar BLS12-381 sig verify	LP-075	Draft
0x0300..22	LP-078	Quasar BLS12-381 aggregate	LP-075	Draft
0x0300..23	LP-078	Quasar Pulsar consensus verify	LP-073	Draft
0x0300..24	LP-078	Quasar hybrid PQ verify	LP-217	Draft
0x0440	LP-078	Bridge gateway	LP-017	Draft
0x0441	LP-078	Cross-chain routing	LP-017	Draft
0x0442	LP-078	Bridge message verification	LP-017	Draft
0x0443	LP-078	Bridge liquidity pools	LP-017	Draft
0x0500..04	LP-078 / LP-126	BLAKE3	LP-011	Draft
0x0500..05	LP-078 / LP-069	Poseidon2	LP-069	Draft
0x0500..06	LP-078 / LP-119	Pedersen (BN254)	LP-119	Draft
0x0500..07	LP-078 / LP-111	Baby Jubjub	LP-111	Draft
0x0500..08	LP-078 / LP-113	Pasta (Pallas + Vesta)	LP-113	Draft
0x0500..10	LP-078	On-chain GraphQL query	LP-128	Draft
0x0500..11	LP-078	GraphQL subscribe	LP-128	Draft
0x0500..12	LP-078	GraphQL cache	LP-128	Draft
0x0500..13	LP-078	GraphQL index	LP-128	Draft
0x0700	LP-078 / LP-066	FHE / TFHE ops	LP-066	Draft
0x0800..02	LP-078 / LP-154	FROST EdDSA threshold	LP-154	Draft
0x0800..03	LP-078 / LP-155	CGGMP21 ECDSA threshold	LP-155	Draft
0x0900	LP-078	Generic ZK verify	LP-037	Draft
0x0901	LP-078 / LP-0120	Groth16 verifier	LP-037	Draft
0x0902	LP-078	PLONK verifier	LP-037	Draft
0x0903	LP-078	fflonk verifier	LP-037	Draft
0x0904	LP-078	Halo2 verifier	LP-037	Draft
0x0A00..01	LP-078 / LP-125	sr25519 (Schnorrkel/Ristretto)	LP-125	Draft
0x3211..00	LP-078 / LP-124	Ed25519	LP-124	Draft
0x3213	LP-078 / LP-131	VRF (ECVRF-Edwards25519-SHA512)	RFC 9381	Draft

Slot	Owner LP	Name	Purpose	Status
0x6010	LP-078	Teleport (cross-chain)	LP-016	Draft
0x9010	LP-078	LXPool – Uniswap v4 PoolManager	LP-032	Draft
0x9011	LP-078	LXOracle – price aggregation	LP-032	Draft
0x9012	LP-078	LXRouter – swap router	LP-032	Draft
0x9013	LP-078	LXHooks – hook registry	LP-032	Draft
0x9014	LP-078	LXFlash – flash accounting	LP-032	Draft
0x9020	LP-078	LXBook – CLOB orderbook	LP-032	Draft
0x9030	LP-078	LXVault – custody + margin	LP-032	Draft
0x9040	LP-078	LXPrice – derived pricing	LP-032	Draft
0x9050	LP-078	LXLend – lending pool	LP-032	Draft
0x9060	LP-078	LXRepayer – self-repaying loans	LP-032	Draft
0x9070	LP-078	LXLiquidator – liquidation engine	LP-032	Draft
0x9080	LP-078	LXTransmuter – debt→collateral	LP-032	Draft
0x9200	LP-078 / LP-122	HPKE seal	LP-122	Draft
0x9202	LP-078	Ring signature verify (LSAG)	—	Draft
0x9203	LP-078 / LP-114	X25519 DH	LP-114	Draft
0x9204	LP-078 / LP-112	Curve25519 raw point ops	LP-112	Draft
0xB002	LP-078 / LP-118	KZG / EIP-4844 blob commitments	EIP-4844	Draft

Table 8: LP-078 band – 53 active rows spanning classical curves, hashing, ZK verifiers, threshold sigs, the LX DEX suite, hybrid KEM, and KZG. Deprecated rows in this band are listed separately (§4.6).

4.2 Federation registry (LP-0011)

4.3 Post-quantum precompile registry (LP-0120)

The canonical PQ precompile band. LP-218 and LP-220 attach P3Q wrappers on top of three of these slots; LP-300 records both the raw-primitive owner (LP-0120 plus the specifying LP) and the wrapper owner where applicable.

Slot	Owner LP	Name	Purpose	Status
0x012201	LP-0120 / LP-072	ML-KEM-768	FIPS 203 KEM (Module-LWE)	Draft

Slot	Owner LP	Name	Purpose	Status
0x012202	LP-0120 / LP-070	ML-DSA single-party	FIPS 204 signature	Draft
0x012203	LP-0120 / LP-071	SLH-DSA-192f	FIPS 205 hash-based sig	Draft
0x012204	LP-0120 / LP-073	Pulsar	Module-LWE threshold (ML-DSA-65 byte-equal)	Draft
0x012205	LP-0120 / LP-218	P3Q / P3Q-Pulsar	Plonky3 PQ-only STARK + LP-218 rollup-batch verifier wrapper	Draft
0x012206	LP-0120 / LP-220	Corona / P3Q-Corona	Ring-LWE threshold; LP-220 wrapper reuses slot	Draft
0x012207	LP-0120 / LP-181 / LP-220	Magnetar / P3Q-Magnetar	SLH-DSA threshold (public-DKG MPC); LP-220 wrapper reuses slot	Draft
0x012208	LP-0120	HQC	NIST PQC4 code-based KEM (family-disjoint from ML-KEM)	Draft
0x012209..0x0122FF	LP-0120	Reserved for PQ extensions	—	Reserved
0x012220	LP-218	Placeholder for displaced Plonky3-fork STARK	Pending dedicated LP allocation	Reserved
0x2221	LP-078 / LP-115	X-Wing	Hybrid KEM (X25519 ‖ ML-KEM-768)	Draft
0x2222	LP-0120 / LP-4930	X-Wing+	Extended hybrid KEM (X25519 ‖ ML-KEM-768 ‖ HQC)	Draft

Table 10: PQ precompile band. The wrapper rows at 0x012205, 0x012206, and 0x012207 share their slot with a raw-primitive owner – the wrapper is dispatched by the body's kind byte and the raw primitive is reachable through the same slot with a different kind. The 0x012220 placeholder is held for a future dedicated-LP allocation of the Plonky3-fork STARK that LP-218 originally targeted there.

4.4 Subnet-EVM-style stateful precompile range (LP-3520)

Slot	Owner LP	Name	Purpose	Status
0x011001	LP-0011	FederationRegistry resolver	Brand sovereignty discovery (LP-0010 family)	Draft
0x011002	LP-0011	BrandConfigStore	Append-only commit log for federation records	Draft
0x011003..0x011FFF	LP-0011	Reserved for federation expansion	—	Reserved

Table 9: Federation registry band owned by LP-0011. Two active slots, one reserve.

Slot	Owner LP	Name	Purpose	Status
0x0200..0001	LP-3520	DeployerAllowList	Contract deployment permissions	Final
0x0200..0002	LP-3520	TxAllowList	Transaction submission permissions	Final
0x0200..0003	LP-3520	FeeManager	Dynamic fee configuration	Draft
0x0200..0004	LP-3520	NativeMinter	Native token minting	Final
0x0200..0005	LP-3520	RewardManager	Validator reward distribution	Final
0x0200..0006	LP-3520	ML-DSA (legacy stateful)	Superseded by LP-0120 0x012202	Deprecated
0x0200..0007	LP-3520	SLH-DSA (legacy stateful)	Superseded by LP-0120 0x012203	Deprecated
0x0200..0008	LP-3520	Warp	Cross-chain BLS messaging	Draft
0x0200..000A	LP-3520	Quasar consensus cert verify	LP-0110 / LP-0120	Draft
0x0200..000B	LP-3520	Corona (legacy stateful)	Superseded by LP-0120 0x012206	Deprecated
0x0200..000C	LP-3520 / LP-154	FROST threshold	LP-154	Draft
0x0200..000D	LP-3520 / LP-155	CGGMP21 threshold	LP-155	Draft

Table 11: Subnet-EVM-style stateful precompile range owned by LP-3520. Three rows are deprecated (0006 ML-DSA, 0007 SLH-DSA, 000B Corona) – superseded by the LP-0120 PQ band; cooldown to 2027-05-18 per §8.

4.5 Reserved future precompile blocks

Slot range	Status	Notes
0x0010..0x00FF	Reserved	Ethereum precompile expansion (held by Ethereum; do not allocate)
0x010000..0x010FFF	Reserved	Future Lux core precompiles
0x013000..0x01FFFF	Reserved	Future PQ precompile extensions
0x020000..0x02FFFF	Reserved	Future stateful precompile blocks (per-VM clusters)

Table 12: Reserved precompile ranges. Allocation only via LP-300 amendment.

4.6 Deprecated precompiles

Six rows currently sit in Deprecated state. They are NOT immediately re-allocatable: §8 pins

Slot	Owner LP	Name	Deprecated	Re-allocatable
0x9201	LP-078 / LP-121	ECIES	2026-05-18	2027-05-18
0x9210	LP-078 / LP-116	AES-256-GCM	2026-05-18	2027-05-18
0x9211	LP-078 / LP-117	ChaCha20-Poly1305	2026-05-18	2027-05-18
0x0200..0006	LP-3520	ML-DSA (legacy stateful)	2026-05-18	2027-05-18
0x0200..0007	LP-3520	SLH-DSA (legacy stateful)	2026-05-18	2027-05-18
0x0200..000B	LP-3520	Corona (legacy stateful)	2026-05-18	2027-05-18

Table 13: Deprecation ledger. All six rows entered the cooldown on 2026-05-18 per AUDIT-2026-05-18 and become re-allocatable on 2027-05-18. The deprecation does not erase the slot; node operators and indexers continue to honor existing on-chain dispatches for the cooldown window.

5 Conflicts resolved by LP-300

LP-300 records – and pins the resolution for – four pre-existing allocation conflicts discovered during the corpus audit. Each row below is a closed conflict; the consuming LPs have been amended (or must be amended) to point at their post-resolution slots. The canonical handoff lives in LP-214 §"Schema ID Allocation" (Option B map); LP-300 reproduces it here as the normative arbiter.

5.1 Conflict 1 – 0xE0..0xEF triple-claim

The single most consequential conflict in the corpus. Three LPs simultaneously claimed bytes inside the same 16-byte block:

- LP-208 originally allocated 0xF0..0xFF for DAG mempool, then reclaimed the prefix to 0xE0..0xE1 after the LP-214 light client claim surfaced.
- LP-211 allocated 0xE2..0xE7 for cross-shard 2PC (six messages).
- LP-218 Draft initially allocated 0xE0..0xEF for rollup envelopes – overlapping LP-208 *and* LP-211.
- LP-214 separately claimed 0xF0..0xF2 for light-client envelopes, colliding with the original LP-208 claim.

Resolution. Per the LP-214 canonical handoff:

LP-219 reuses LP-211's 0xE2..0xE7 at the rollup tier; this is treated as a ShapeKind-level extension (TxKind 0x44 inside the LP-218 0xEC envelope) rather than a TypeKind-level reclamation, so it does not contend for the slot at namespace 1.

5.2 Conflict 2 – LP-218 P3Q slot vs displaced Plonky3 STARK

LP-218 attaches a P3Q wrapper at PQ precompile slot 0x012205. The original LP-218 Draft cited a Plonky3-fork STARK as the underlying primitive at the same slot. LP-0120's PQ precompile registry instead pins 0x012205 to the P3Q rollup-batch verifier wrapper, with the raw STARK primitive displaced.

LP	Final range	Rationale
LP-208	`0xE0..0xE1`	Earliest concrete Draft with measurable wire impact; takes the low edge of the block.
LP-211	`0xE2..0xE7`	Six-message contiguous block; sat unconflicted at original allocation.
LP-218	`0xE8..0xEF`	Shifted from prior `0xE0..0xE3` claim; gets the contiguous 8-byte tail of the block.
LP-214	`0xF0..0xF2`	Sits in the framing future range; LP-208 vacated `0xF0..0xF1` when it moved to `0xE0..0xE1`.

Table 14: Final `0xE0..0xFF` layout. Every consuming LP must amend its "Wire schemas" section to cite the post-resolution range and reference LP-300 §2 as the binding map.

Resolution. `0x012205` is owned by LP-0120 with the LP-218 wrapper. The displaced Plonky3-fork STARK is held at the placeholder slot `0x012220` pending a dedicated LP allocation. LP-300 records both states: `0x012205` is live for the wrapper; `0x012220` is Reserved for the displaced primitive. The wrapper's verifier dispatches the underlying primitive by kind byte at the body level, so a future move of the raw primitive to a different slot does not break LP-218.

5.3 Conflict 3 – LP-220 sibling-slots vs LP-218 kind-dispatch

LP-220 proposed allocating sibling slots `0x012206` (Corona) and `0x012207` (Magnetar) as wrapper-only entries, distinct from LP-218's slot reuse at `0x012205`. The conflict surfaced when LP-0120's PQ registry assigned the same two slots to the raw primitives – Corona at `0x012206` (Ring-LWE threshold) and Magnetar at `0x012207` (SLH-DSA threshold).

Resolution. LP-220 wrappers share their slot with the raw primitives, dispatched by kind byte at the body level – the same pattern LP-218 uses at `0x012205`. LP-0120 keeps `0x012206` and `0x012207` as the raw-primitive allocation; LP-220 attaches the P3Q-Corona and P3Q-Magnetar wrappers on the same slots. LP-0120's raw Groth16 verifier at `0x0901` remains separate. The wrapper-vs-primitive ambiguity is resolved purely at the body kind byte, never at the slot.

5.4 Conflict 4 – Deprecated allocations entering cooldown

Six allocations entered Deprecated state during the 2026-05-18 precompile sweep (AUDIT-2026-05-18). A Deprecated slot is NOT immediately re-allocatable: the slot remains nominally owned by its deprecating LP for the cooldown window, during which it MUST NOT be reused by a different LP.

Resolution. §8 pins the cooldown policy. The six rows become re-allocatable on 2027-05-18: `0x9201`, `0x9210`, `0x9211`, `0x0200..0006`, `0x0200..0007`, and `0x0200..000B`. The deprecation entries persist in LP-300 across the cooldown to prevent accidental re-use.

5.5 Resolution authority and amendment chain

For each closed conflict the consuming LPs MUST amend their relevant "Wire schemas" / "Precompile slot" / "Schema-ID allocation map" section to:

1. State the final post-resolution slot or range.
2. Reference LP-300 §2 (TypeKind) or §4 (precompile) as the binding map.
3. Drop any duplicated allocation tables; the consuming LP becomes informative-only on the allocation question.

A consuming LP at status Final without a matching LP-300 row, or with a row that disagrees with LP-300, is itself in violation and must be amended in the same PR cycle.

6 Conflict resolution policy

LP-300 is the single source of truth. Any future schema ID, TypeKind byte, ShapeKind byte, or EVM precompile slot allocation MUST be added here first; the consuming LP MUST reference (not duplicate) the assignment.

6.1 Adding a new allocation

1. The proposer opens a PR against LP-300 listing the requested slot(s), the consuming LP number, and a one-line purpose. The PR cites the namespace (TypeKind / ShapeKind / Precompile) and the authoritative section in this paper that the new row will join.
2. The PR is merged BEFORE the consuming LP can be promoted past "Draft" status. A consuming LP at "Final" without a matching LP-300 row is itself in violation.
3. The consuming LP's "Wire schemas" or "Precompile slot" section references LP-300 by row; it does not re-state the allocation. The consuming LP becomes informative-only on the allocation question.

6.2 Resolving overlap

If two LPs claim overlapping ranges:

1. The LP-300 maintainer arbitrates. The default rule: the chronologically earlier "Final" claim wins; the later claim relocates to the next free range. Pre-activation, neither has shipped wire bytes – the call is editorial.
2. The losing LP is amended to point at its new range, and an "Activation marker" note is added to LP-300 stating "Was `<old range>`; moved to `<new range>` per LP-300 §5."
3. Pre-activation (before unix `1766708400`), reshuffles are free because no production wire bytes have shipped. Post-activation, reshuffles require a network upgrade.

6.3 Maintainer authority

The LP-300 maintainer is the editor of last resort for any allocation conflict. The maintainer's role is to record decisions in this paper; the decisions themselves come from the technical leads of the consuming LPs. When the consuming LPs cannot agree, the maintainer applies the "earliest Final wins" rule from above.

The four conflicts already resolved (§5) were arbitrated by the LP-300 maintainer in 2026-05 – 2026-06. All four resolutions are now Final.

6.4 Wire bytes and the activation cliff

Pre-activation (before `1766708400`), LP-300 is a planning document and reshuffles cost nothing. The four conflicts in §5 were all resolved pre-activation; no shipped wire byte changed semantics.

Post-activation, every byte allocated in LP-300 is interpreted by some live validator somewhere. A reshuffle past this point is a network upgrade, with all the operational cost that implies: coordinated node rollout, dual-decode windows for the old and new allocations, and an explicit deprecation cooldown (§8) on the old slot. Post-activation reshuffles should be rare.

6.5 Out-of-band edits forbidden

A consuming LP MUST NOT silently change an allocation it owns. The canonical procedure is: open the LP-300 amendment PR first, get it merged, then propagate the change to the

consuming LP. Even a typographic correction to an allocation row goes through the LP-300 amendment process to preserve the audit trail.

7 Cross-referencing pattern

Every consuming LP's "Wire schemas" / "Schema IDs" / "Precompile slot" section is now informative-only. LP-300 is normative. The canonical pattern a consuming LP MUST use, with the wording lightly adapted to taste:

```
This LP allocates the following schema IDs (registered in
LP-300 sec.Master Table):

  - 0xD0 ConsensusVote
  - 0xD1 ConsensusProposal
  - 0xD2 ConsensusCert (mirrors 0x01 QuasarCert)
  - ...

This LP's "Wire schemas" section is informative only;
LP-300 is normative. For the master allocation map see
LP-300 sec.Master Table (TypeKind / ShapeKind /
Precompile slot).
```

The pattern has three load-bearing properties:

1. **List the rows the LP owns, but cite LP-300 for the table.** The consuming LP's reader needs to know which bytes are in play; the consuming LP does not need to duplicate the master registry to deliver that knowledge.
2. **Mark the local section informative.** If the consuming LP and LP-300 ever disagree – a typo, an out-of-date Draft, a late-arriving amendment that hasn't propagated – the disagreement is resolved in favor of LP-300. The "informative only" label is the operator-facing surface of that rule.
3. **Cite the binding map by section.** A reader who reaches the consuming LP via search lands on the local list, then follows the LP-300 cite to the binding map. The local list never grows stale beyond the LP-300 PR cycle.

7.1 What consuming LPs MUST NOT do

- Re-state the master table. A consuming LP that mirrors the rows owned by other LPs is a maintenance liability; every LP-300 amendment would have to chase every consuming-LP mirror.
- Allocate a slot not in LP-300. A consuming LP that names a byte LP-300 doesn't have is making a unilateral claim. The amendment-first rule (§6) forbids this.
- Silently change an allocation. Edits to the local row must follow an LP-300 amendment, not lead it.

7.2 What consuming LPs SHOULD do

- Inline a short rationale next to the local row. "LP-300 row: `0xE8` `RollupBatchTx` – chosen because the contiguous `0xE8..0xEF` tail leaves room for sequencer-rotation and cross-rollup-commit additions in `0xED..0xEF`."
- Cite the conflict resolution row (§5) when the LP's allocation was the result of a conflict resolution. Future readers need the audit trail.
- Reference the namespace explicitly. "LP-300 namespace 1 (TypeKind)" is unambiguous; "LP-300 row `0x9010`" is ambiguous because the same hex prefix could appear in multiple namespaces if not qualified.

7.3 Audit checklist for the consuming LP

Before promoting a consuming LP past Draft:

1. Every slot the LP names appears in LP-300 with the same Owner LP. (Cross-check against §2, §3, §4.)
2. Every wrapper relationship (LP attaches to a slot owned by a different LP) is recorded in LP-300 with both LPs cited in the Owner LP column.
3. The consuming LP's local table is labeled informative and cites LP-300 as the binding map.
4. Any allocation change since the previous LP revision has a matching LP-300 amendment PR that was merged before the consuming LP's promotion PR.

A consuming LP that fails any of these checks is not ready to promote past Draft.

8 Cooldown for deprecated allocations

When an allocation is marked Deprecated, the slot is NOT immediately reusable. A one-year cooldown applies before re-allocation, to give node operators and indexers time to drop the old code paths.

8.1 Why a cooldown

Three concrete failure modes if a deprecated slot is re-allocated without cooldown:

1. **Stale node software.** Validator binaries pinned to the pre-deprecation release continue to dispatch the old precompile. A re-allocation makes that dispatch return semantically-different data; the validator's view of the chain forks silently from the network.
2. **Indexer leaks.** Off-chain indexers that scrape historical data may not pick up the new allocation immediately. Returning historical traces conflates the old and new semantics on the same slot.
3. **Audit confusion.** Forensic analysis of a contract that called the deprecated slot becomes ambiguous: was the dispatch pre- or post-re-allocation? Without a cooldown window, the line is blurred.

The cooldown removes all three failure modes by guaranteeing a calendar window in which the slot is dead – no dispatch is valid, no allocation is honored.

8.2 The current cooldown ledger

Six allocations entered Deprecated state on 2026-05-18 per AUDIT-2026-05-18 (the precompile sweep). All six become re-allocatable on 2027-05-18.

8.3 Cooldown procedure

1. An LP-300 amendment moves the row from its active table to the Deprecated table (§4.6 for precompiles; analogous tables for TypeKind / ShapeKind).
2. The amendment records both the `Deprecated` date and the `Re-allocatable` date (Deprecated + 365 days).
3. The consuming LP that owned the deprecated slot is amended in the same PR cycle to mark the deprecation and cite the LP-300 row.
4. During the cooldown, no LP MAY re-allocate the slot. A PR that attempts to do so is rejected by the LP-300 maintainer until the re-allocatable date has passed.
5. On or after the re-allocatable date, the slot becomes candidate for re-allocation per the standard amendment-first rule (§6). The LP-300 row stays in the Deprecated table until a new amendment moves it elsewhere.

Slot	Owner LP	Reason	Re-allocatable
0x9201	LP-078 / LP-121	ECIES superseded by HPKE (0x9200) and X-Wing/X-Wing+ hybrid KEMs	2027-05-18
0x9210	LP-078 / LP-116	AES-256-GCM moved out of EVM precompile (use stdlib AEAD)	2027-05-18
0x9211	LP-078 / LP-117	ChaCha20-Poly1305 moved out of EVM precompile (use stdlib AEAD)	2027-05-18
0x0200..0006	LP-3520	ML-DSA stateful slot superseded by LP-0120 0x012202 (FIPS 204 verify)	2027-05-18
0x0200..0007	LP-3520	SLH-DSA stateful slot superseded by LP-0120 0x012203 (FIPS 205 verify)	2027-05-18
0x0200..000B	LP-3520	Corona stateful slot superseded by LP-0120 0x012206 (Ring-LWE threshold)	2027-05-18

Table 15: The six rows currently in cooldown. Until 2027-05-18 they sit in the Deprecated bucket of LP-300; on 2027-05-18 they become candidates for re-allocation, subject to the amendment-first rule (§6). Re-allocation does not have to happen on 2027-05-18; the slot can stay reserved indefinitely after the cooldown ends.

8.4 Activation marker

LP-300 itself is **Final** from genesis – unix 1766708400 = 2025-12-25T16:20:00-08:00 – because it is a meta-registry, not a wire change. The first time the consuming LPs (LP-208, LP-211, LP-214, LP-218, LP-219, LP-220) cross activation in production, LP-300 is the binding map. Any disagreement between this LP and a consuming LP is resolved in favor of LP-300; the consuming LP must be amended in a follow-up PR.

The activation marker is also the reference time-zero for cooldown arithmetic. A deprecation on 2026-05-18 (pre-activation) carries forward post-activation as a calendar window, not a height window; the cooldown is measured in days from the deprecation date, not blocks from activation.

References