Opstan (OPST)

Whitepaper / Consensus Rules

Complete specification of consensus rules, limits and constraints of the network (EN).

Document date: 2026-02-22
Source: source code (archive tito.zip)

Network Metadata

Document date: 2026-02-22
Source: source code (archive tito.zip)

Field	Value
NETWORK_ID	0x4F535054
Genesis (timestamp)	1754524800 (2025-08-07 00:00:00 UTC)
Genesis header hash	134dc03a782a442cab7264565f3d97d3d5cf10aba2bdf516490da8140277ac52
FREEZE (MAINNET_CONSENSUS_LOCK)	v2-38704739-75bd301b7482a2c1dc647cbe5572a5b4b83c6a7a8cb2efbfd961c654ea6ece3e
Deps fingerprint (expected)	deps1-53dd0f96518e7ec25fea7161cd7dc396e5f96ffa296da3543da38a66f7a098a3
P2P protocol version	6

Sections marked **Consensus** define validity of blocks and on-chain objects. Sections marked **Node policy** define local mempool/network limits and do not affect consensus.

Contents

1. Terms and notation

1 OPST = 10^18 atoms (Atoms, u128). **ATOMS_PER_OPST** = 1000000000000000000.

H256 is a 32-byte sha256d hash. **Wire V1** is a binary serialization format for on-chain objects.

2. Network identity and genesis

NETWORK_ID: 0x4F535054. **GENESIS_TIME**: 1754524800 (2025-08-07 00:00:00 UTC).

GENESIS_HEADER_HASH_HEX: 134dc03a782a442cab7264565f3d97d3d5cf10aba2bdf516490da8140277ac52.

2.1 FREEZE

MAINNET_CONSENSUS_LOCK must match the computed consensus fingerprint; otherwise the node will refuse to start.

```
v2-38704739-75bd301b7482a2c1dc647cbe5572a5b4b83c6a7a8cb2efbfd961c654ea6ece3e
```

3. Block format and hashing

PoW hash: **sha256d(encode_pow_v1(header))**. **encode_pow_v1** length: 252 bytes.

extra: 32 ASCII bytes in the range [0-9a-z] (consensus).

Object ID: **sha256d([0x01] || canonical_encode_v1)**. Merkle roots are recomputed and verified; duplicates are forbidden.

4. PoW and target

A block is valid if **hash <= target** (big-endian comparison).

The target may not be easier than **MIN_TARGET**.

```
MIN_TARGET: 00000000ffff0000000000000000000000000000000000000000000000000000
```

Version rules: top bits must satisfy the version mask, and additionally **version == BASE_VERSION** (strict).

5. Time

MTP is the median of timestamps of the last 11 blocks (MTP_WINDOW=11).

- **timestamp > parent_mtp** (strict).
- **timestamp <= adjusted_now + 7200** (max future drift 2 hours).

6. Retarget

BLOCK_TARGET_SECS=600. **RETARGET_INTERVAL**=144. **span**=86400.

At the window boundary, elapsed is clamped to the range 9/10..11/10 of span. Within the window: **target == parent.target**. At the boundary: **target must exactly equal the expected value** (strict).

7. Size limits (consensus)

Parameter	Value
MAX_TX_BYTES	1001
MAX_MSG_BYTES	1201
MAX_PRIVMSG_BYTES	1601
MAX_SUM_TX_BYTES	4194304
MAX_SUM_MSG_BYTES	3145728
MAX_SUM_PRIVMSG_BYTES	2097152
MAX_BLOCK_TOTAL_BYTES	10485760

The total limit includes sum(tx)+sum(msg)+sum(priv)+252 and must not exceed 10 MB.

8. Block application (balances, nonce, coinbase)

A single nonce space is shared across msg/tx/privmsg. Order within a block: **msgs** → **transfer txs** → **priv_msgs**.

For each address, the expected nonce is **stored_nonce + 1**; after acceptance the nonce increments.

Fees (**fee**) are summed across all objects in a block and must match **coinbase.fees**.

• **TransferTx**: amount > 0; deduct amount+fee; recipient receives amount.
• **PlainMsg/PrivMsg**: deduct only fee.
• Balance cannot become negative; u128 overflows are forbidden.

CoinbaseTx (first in txs) must match strictly: height==header.height; miner==header.miner; reward==coinbase_reward(height); fees==Σfee.

Reward maturation: reward+fees are credited at height+100 (**COINBASE_MATURITY**).

9. On-chain objects and signatures

TransferTx / PlainMsg / PrivMsg are signed with Ed25519 over a network-bound V2 preimage:

```
u16_le(len(domain)) || domain || NETWORK_ID_LE(4) || encode_v1(body)
```

Channel name (PlainMsg): length 1..32, only [a-z0-9].

10. Emission

Phase	Length (blocks)	Reward (OPST / block)
1	52 560	240
2	210 240	120
3	210 240	60
4	210 240	30
5	210 240	15
6	210 240	8
7	210 240	4
8	210 240	2
9	∞	1

After the final phases: 1 OPST per block forever (no total emission cap).

11. Channels: ownership

The first post claims a channel. Afterwards, only the owner can post. Ownership transfer is specified by a payload of the form **owner:<64hex>**.

12. Chain selection (chainwork)

work = floor((2^256 - 1)/(target+1)) + 1; the branch with the maximum cumulative work is selected.

```
work = floor((2^256 - 1) / (target + 1)) + 1
```

13. Node policies (non-consensus)

Parameter	Value
MEMPOOL_CAP	80000
MEMPOOL_PER_ADDR_MAX	50
MEMPOOL_TTL_BLOCKS	12

P2P Hello verifies: protocol version=6, NETWORK_ID+genesis, consensus_lock and deps_hash.

Appendix A. Constants (src/params.rs)

Constants are listed exactly as in the source (names preserved).

Constant	Value
BASE_VERSION	VERSION_TOP_BITS
BLOCK_TARGET_SECS	600
CHANNEL_NAME_MAX	32
COINBASE_MATURITY	100
DECIMALS	18
GENESIS_HEADER_HASH_HEX	"134dc03a782a442cab7264565f3d97d3d5cf10aba2bdf516490da8140277ac52"
GENESIS_TIME	1_754_524_800
HEADER_ENCODED_LEN	252
MAX_BLOCK_TOTAL_BYTES	10 * 1024 * 1024
MAX_FUTURE_DRIFT_SECS	2 * 3600
MAX_MSG_BYTES	1201
MAX_PRIVMSG_BYTES	1601
MAX_SUM_MSG_BYTES	3 * 1024 * 1024
MAX_SUM_PRIVMSG_BYTES	2 * 1024 * 1024
MAX_SUM_TX_BYTES	4 * 1024 * 1024
MAX_TX_BYTES	1001
MEMPOOL_CAP	80_000
MEMPOOL_PER_ADDR_MAX	50
MEMPOOL_TTL_BLOCKS	12
MIN_TARGET	H256([00 00 00 00 ff ff 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00])
MTP_WINDOW	11
NETWORK_ID	0x4F53_5054
RETARGET_INTERVAL	144
RETARGET_STRICT_STEP_MAX_DEN	10
RETARGET_STRICT_STEP_MAX_NUM	11
RETARGET_STRICT_STEP_MIN_DEN	10
RETARGET_STRICT_STEP_MIN_NUM	9
RETARGET_WINDOW_SECS	BLOCK_TARGET_SECS * RETARGET_INTERVAL
VERSION_TOP_BITS	0x20_00_00_00
VERSION_TOP_MASK	0xE0_00_00_00

Appendix B. Key constants (p2p.rs) — policy

These limits affect networking behavior but are not part of consensus.

```
const PROTOCOL_VERSION: u32 = 6;
const FRAME_HEADROOM_BYTES: usize = 256 * 1024;
const MAX_WIRE_FRAME_BYTES: usize = (MAX_BLOCK_TOTAL_BYTES as usize) +
FRAME_HEADROOM_BYTES;
const PRE_HANDSHAKE_MAX_FRAME_BYTES_DEFAULT: usize = 64 * 1024;
const IO_READ_TIMEOUT: Duration = Duration::from_secs(180);
const CONNECT_RETRY_MIN: Duration = Duration::from_secs(2);
const CONNECT_RETRY_MAX: Duration = Duration::from_secs(20);
const RECONNECT_BACKOFFS_DEFAULT: [u64; 5] = [30, 60, 120, 300, 3000];
const CONNECT_TIMEOUT_DEFAULT_SECS: u64 = 8;
const TIP_INTERVAL: Duration = Duration::from_secs(10);
const PING_INTERVAL: Duration = Duration::from_secs(30);
const PEERS_GOSSIP_INTERVAL: Duration = Duration::from_secs(60);
const MAX_GOSSIP_PEERS: usize = 512;
const MAX_PEER_STR: usize = 128;
const MAX_PEERS_FROM_MSG: usize = 512;
const PEERS_FILE_MAX: usize = 10_000;
const PEERS_DELETE_ON: usize = 8_000;
const PEERS_DELETE_OFF: usize = 6_000;
const PRUNE_FAIL_THRESHOLD: u8 = 2;
const PEERSNC_FILE_MAX: usize = 100_000;
const MAX_TX_WIRE_BYTES: usize = 1024 * 1024;
const MEMPOOL_REQ_INTERVAL: Duration = Duration::from_secs(90);
const MEMPOOL_SNAPSHOT_MAX_ITEMS_DEFAULT: usize = 256;
const MEMPOOL_SNAPSHOT_MAX_BYTES_DEFAULT: usize = 256 * 1024;
const MEMPOOL_SYNC_MAX_BEHIND_DEFAULT: u64 = 2000;
const UPNP_LEASE_SECS: u32 = 3600;
const UPNP_REFRESH_EVERY: Duration = Duration::from_secs(25 * 60);
const MAX_OUTBOUND_TOTAL_DEFAULT: usize = 64;
const MAX_INBOUND_TOTAL_DEFAULT: usize = 32;
const MAX_INBOUND_PER_IP_DEFAULT: usize = 16;
const OUTBOUND_SLOT_WAIT: Duration = Duration::from_secs(1);
const OUTBOUND_TASKS_CAP_DEFAULT: usize = 256;
```