SN98 ForeverMoney - Tech Doc

Summary

As liquidity moves on-chain, SN98 aims to become the leading decentralized Automated Liquidity Manager (ALM). Powered by a network of miners predicting optimal liquidity strategies, SN98 maximizes trading efficiency and fee generation while minimizing inventory risk - evolving into a decentralized market-making platform.

System Overview

SN98 consists of four primary subsystems:

1. **Miners** - propose LP strategies and may contribute liquidity to SN98 vaults.
2. **Validators** - provide pool data, score miner strategies, rank miners, and publish winning strategies.
3. **Subnet Owner** - provides validator code, takes the top-performing strategy, converts it into v3 NFT LP operations via a multisig-controlled **executor** system.
4. **Vaults** - hold and deploy liquidity according to LP strategies. Distribute fees to vault owner and fees value into the Alpha token economy.

Miners

Miners perform two primary functions:

(a) Strategy Proposers

They submit:

- Tick ranges
- Allocation sizes
- Rebalance rules
- Metadata or confidence values
- Strategies must comply with validator constraints

(b) LP Participants

Miners may deploy liquidity vaults which are managed by the subnet.

Validators

Validators evaluate miners along **two scoring dimensions**:

I. Net PnL vs HODL (70%)

- Backtested performance
- Compared to a passive HODL baseline
- **Top-heavy scoring** — only the top 3 strategies receive full weight

II. LP Fee Share of Miners (30%)

- Measures how much fees where generated from miners vaults
- Scored **pro-rata**

Final Score = 70% Performance + 30% LP Alignment

Validators also enforce constraints (IL caps, minimum tick widths, rebalance limits) and publish rankings, diagnostics, and competition parameters. Scoring weights may be adjusted as needed.

2.3 Subnet Owner — Executor Bot

Responsible for:

- Reading the winning strategy
- Converting it into v3 NFT LP ticks
- Executing rebalances or creating new positions
- Reporting execution results

MVP: **manual execution** via multisig.
Future: **fully automated execution**.

2.4 Vaults

Vaults hold SN98-managed liquidity.

Types of vaults:

1. **CreatorBid Trenches Vaults** - hold LP raised on the CreatorBid launchpad; SN98 manages ranges and takes a fee.
2. **SN98 Vaults** - allow deposits and withdrawals:
    - **Whitelisted vaults:** only specific participants may deposit or withdraw.
    - **Public vaults:** can be created by anyone for any pair, allowing open LP top-ups.

Fee Model

Fee revenue split:

- **30% total fee take on LP fees**, sent to the SN98-managed Safe wallet:
  - 50% → Alpha buyback
  - 50% → Research & Development

---

3. Interaction Diagram (Text Flow)

1. Validator publishes round parameters.
2. Miners generate and submit strategies.
3. Validators backtest and score strategies.
4. Executor bot deploys the winning strategy.
5. Vaults earn fees and distribute revenue.

4. Miner Data Model & API Format

Miners & Validators interact with each other through a standardized data-exchange model. This ensures validators can reliably query miner endpoints, provide consistent inputs, and receive structured strategy predictions for specific blocks.

4.1 Data Provided to Miners

Validators send miners all required contextual data to compute a recommended liquidity position. Miners do not need to fetch their own chain data — instead, they can rely on validator-provided inputs and the shared Postgres database. However, they are free to use any source they see fit, ex. their own DB, subgraph etc.

Miners receive:

(a) Trading Pair

The validator request specifies the trading pair for which the miner must propose a strategy.

Example:

```
"pairAddress": “0x0000000000000000000000000000000000000000”
```

Although miners may hard-code pairs during early phases, the production design assumes the pair is part of the request.

(b) Public Postgres Database Access

Miners are given access to a *read-only* Postgres instance.

Properties:

- Public or lightly authenticated via simple credentials
- Contains exclusively public on-chain data
  Includes *all pool events* for the requested trading pair
  (swaps, mints, burns, fee growth, price updates, tick changes, etc.)
- May include events for other pools, but miners can assume all relevant data is available

This allows miners to run:

- Their own backtests
- Internal simulations
- Confidence scoring

(c) Inventory or Current Position

Validators provide one of the following:

1. **Inventory Mode (default)**

Miner receives:

```
"inventory": {
    "amount0": "...",
    "amount1": "..."
}
```

    - Miner decides how to deploy these assets into v3 positions.

2. **Existing Position Mode**

If a vault is already deployed, validator may instead provide the *current* set of active positions:

```
"current_positions": [
    {
        "tickLower": ...,
        "tickUpper": ...,
        "liquidity": "..."
    }
]
```

- Miner proposes adjustments or rebalances.

(d) Block Number (Prediction Target)

Miners are given a **block number** and must return a recommended configuration *as if the vault were to rebalance exactly at that block*.

Example:

```
"target_block": 12481234
```

The prediction is forward-looking relative to the provided datasets.

4.2 Miner Endpoint Requirements

Each miner must host an HTTP endpoint (public or authenticated) that validators can query frequently.

Endpoint Responsibilities:

- Accept strategy-prediction requests
- Process provided inputs, database queries, and internal models
  Return a full v3 LP strategy for the target block

Availability Expectations

- High availability (validators may poll often)
- Stateless (each request fully describes the inputs)

4.3 JSON API Format

Below is the reference model for validator → miner communication.

4.3.1 Request Format

```
{
  "pairAddress": “0x0000000000000000000000000000000000000000”,
  “chainId”: 8453,
  "target_block": 12345678,
  "mode": "inventory",
  "inventory": {
    "amount0": "1000000000000000000",
    "amount1": "2500000000"
  },
```

```
  "current_positions": [],
  "metadata": {
    "round_id": "2025-02-01-001",
    "constraints": {
      "max_il": 0.10,
      "min_tick_width": 60,
      "max_rebalances": 4
    }
  }
}
```

Notes:

- `mode` can be `"inventory"` or `"position"`.
- If `mode = "position"`, then `current_positions` will be populated and `inventory` may be omitted.

4.3.2 Response Format (Miner → Validator)

```
{
  "strategy": {
    "positions": [
      {
        "tickLower": -9600,
        "tickUpper": -8400,
        "allocation0": "500000000000000000",
        "allocation1": "0",
        "confidence": 0.82
      },
      {
        "tickLower": -8400,
        "tickUpper": -7200,
        "allocation0": "0",
        "allocation1": "1800000000",
        "confidence": 0.78
      }
    ],
    "rebalance_rule": {
      "trigger": "price_outside_range",
```

```
      "cooldown_blocks": 300
    }
  },
  "miner_metadata": {
    "version": "1.0.0",
    "model_info": "lstm-v3-swaps-optimized"
  }
}
```

Interpretation:

- `positions` defines how liquidity should be placed at the target block.
- `confidence` is optional but useful for ranking diagnostics.
- `rebalance_rule` is optional — miners may leave it empty if unnecessary. Validators convert this into concrete v3 NFT operations.

4.4 Miner Strategy Model

All miner logic is encapsulated in the prediction endpoint.
Validators do not require miners to submit separate documents, metadata, or strategy packages.

A “strategy” in SN98 is defined as:

“A function that maps `(pair, block, inventory | current_position, events)` → `set of v3 liquidity positions`.”

Miners are fully free to:

- Use ML models
    - Use rule-based systems
- Use hybrid predictive agents
- Query historical data from Postgres

Validators only care about the *output format* and *performance*.