

ZERO OS — Self-Protection Architecture

Scenario: The signal API (external dependency) becomes unavailable. License expires, tokens run out, API provider changes terms, relationship ends, service shuts down.

Requirement: ZERO OS must NEVER leave a user stranded. It must protect open positions, degrade gracefully, and continue operating with reduced capability. The OS survives the loss of any single dependency. **Principle:** ZERO OS is the OS. The signal API is one app running on it. If the app dies, the OS keeps running.

THREAT MODEL

Scenario	Likelihood	Speed	Impact
x402 balance runs out	HIGH	Gradual (predictable)	Signal quality drops to basic
API rate limited / throttled	MEDIUM	Sudden	Fewer signals per cycle
API subscription/license expires	MEDIUM	Known date	Full signal loss
API returns errors / goes down	MEDIUM	Sudden	No new signals
API provider changes terms	LOW	Weeks notice	May need to renegotiate
API provider shuts down entirely	LOW	Months notice	Permanent signal loss
Relationship with API provider ends	LOW	Days-weeks	Need alternative signal source

PROTECTION LAYER 1: GRACEFUL DEGRADATION

The system has FOUR operating modes based on signal availability:

MODE 1: FULL INTELLIGENCE

Signal API: connected, paid, all endpoints available
Behavior: normal operation, full signal scoring,
strategy assembly, portfolio optimization
Dashboard shows: ● FULL

MODE 2: CACHED INTELLIGENCE

Signal API: unavailable, but recent data cached locally
Behavior: uses last known signals (max cache age: 4 hours)
No new entries. Existing positions managed normally.
Immune system continues. Stops remain on HL.
Dashboard shows: ● CACHED (Xs stale)

MODE 3: BASIC INTELLIGENCE

Signal API: unavailable beyond cache window OR tokens depleted
Behavior: falls back to LOCAL signal generation
Basic indicators (RSI, EMA, MACD) computed locally
from free price data (HL WebSocket is free, no API needed)
Reduced quality but still functional
Existing positions managed. New entries use basic signals.
Dashboard shows: ● BASIC (degraded)

MODE 4: PROTECTION ONLY

Signal API: gone. Local signals: disabled or user chose to stop.
Behavior: NO new entries.
Existing positions managed to completion (stops, exits, max hold)
Immune system runs at full capacity
System winds down positions safely over time
Dashboard shows: ● PROTECTING (winding down)

Implementation

```
class SignalMode:
    FULL = "full"           # API available, all features
    CACHED = "cached"       # Using last known signals
    BASIC = "basic"         # Local indicators only
    PROTECTION = "protection" # No new trades, manage existing

class SignalManager:
    def __init__(self):
        self.mode = SignalMode.FULL
        self.cache = {}      # last signals per coin
        self.cache_timestamp = None
        self.x402_balance = None
        self.max_cache_age = 4 * 3600 # 4 hours

    def get_signals(self, coins):
```

```

# Try API first
try:
    signals = self.call_api(coins)
    self.cache = signals
    self.cache_timestamp = time.time()
    self.mode = SignalMode.FULL
    return signals
except PaymentRequired:
    # x402 balance depleted
    self.mode = SignalMode.BASIC
    return self.get_basic_signals(coins)
except APIError:
    # API down or unavailable
    if self.cache_is_fresh():
        self.mode = SignalMode.CACHED
        return self.cache
    else:
        self.mode = SignalMode.BASIC
        return self.get_basic_signals(coins)

def cache_is_fresh(self):
    if not self.cache_timestamp:
        return False
    return (time.time() - self.cache_timestamp) < self.max_cache_age

def get_basic_signals(self, coins):
    """
    Generate basic signals from FREE data sources.
    HL WebSocket provides price data for free.
    Compute RSI, EMA, MACD locally.
    Lower quality than API signals but functional.
    """
    # This is the ZERO OS safety net
    # It doesn't need the API to work
    # It just works worse without it
    pass

```

PROTECTION LAYER 2: LOCAL SIGNAL CACHE

Every signal response from the API is cached locally.

```

~/.zeroos/cache/
├─ signals/
|   └─ BTC_latest.json      # last signal scores

```

```

|   ├── ETH_latest.json
|   └── ...
├── strategies/
|   ├── BTC_strategy.json    # last assembled strategy
|   └── ...
├── portfolio/
|   └── optimization.json    # last portfolio weights
└── metadata.json           # timestamps, API version, cache health

```

Cache Rules

1. Every successful API response is cached to disk immediately
2. Cache includes timestamp of when data was received
3. Cache is NEVER deleted by the system (only by explicit user action)
4. On startup, load cache before calling API
5. If API is down on first startup, system runs from cache
6. Cache age is displayed in dashboard: "Signals: 2h old (cached)"

CACHE LIFETIME RULES:

- 0-1 hour: signals are "fresh" – trade normally
- 1-4 hours: signals are "aging" – reduce position sizes by 50%
- 4-12 hours: signals are "stale" – no new entries, manage existing
- 12+ hours: signals are "expired" – protection mode, wind down

What Gets Cached (Everything)

- ✓ Signal scores per coin (Sharpe, WR, direction, stop_pct)
- ✓ Assembled strategies per coin
- ✓ Portfolio optimization weights
- ✓ Regime detection results
- ✓ Indicator values (for basic signal fallback computation)
- ✓ Blacklist state
- ✓ Universe membership

This means if the API disappears PERMANENTLY,
ZERO OS has the last known state of every signal
and can make informed decisions about winding down.

PROTECTION LAYER 3: LOCAL BASIC SIGNAL ENGINE

ZERO OS ships with a BUILT-IN basic signal engine that works WITHOUT any external API.
This is the "basic plugin" concept from the original system spec — but now it's a survival

mechanism, not just a free tier.

LOCAL SIGNALS (computed from free HL data):

1. Price data: HL WebSocket provides free real-time prices
No API key needed. No payment needed. Always available.
2. Basic indicators computed locally:
 - RSI (14-period)
 - EMA crossover (9/21)
 - MACD
 - Bollinger Bands
 - Volume profile
 - Funding rate (from HL API, free)
3. Basic signal logic:
 - RSI < 30 + EMA cross up = LONG signal
 - RSI > 70 + EMA cross down = SHORT signal
 - No Sharpe scoring, no backtesting, no optimization
 - Quality: 5/10 compared to API signals (10/10)
 - But it WORKS and it's FREE
4. When to use:
 - x402 balance = 0 and no cache available
 - API permanently unavailable
 - User explicitly chooses basic mode (free tier)

Why This Matters for IP Protection

The basic signal engine means ZERO OS is a COMPLETE product without the signal API. It's degraded, but functional.

If the relationship with the API provider ends:

- Users don't lose their money (positions protected)
- Users don't lose their software (ZERO OS still runs)
- Users don't lose ALL signal intelligence (basic engine works)
- You have TIME to find or build an alternative signal source
- The OS layer (immune system, multi-agent, reasoning, dashboard) continues operating at 100% capacity

ZERO OS's IP is:

- The immune system (13 checks, self-audit, desync detection)
- The multi-agent orchestrator
- The Claude reasoning engine
- The dashboard and terminal aesthetic

- The CLI and user experience
- The basic signal engine

NONE of these depend on the external API.

The API makes them BETTER. It doesn't make them EXIST.

PROTECTION LAYER 4: x402 BALANCE MONITORING

The system proactively monitors x402 balance and warns before it runs out.

x402 BALANCE STATES:

HEALTHY: balance > 7 days of estimated usage
 No action needed.

WARNING: balance = 3-7 days remaining
 Dashboard: "Signal credits running low. Top up to
 maintain full intelligence."
 Agent continues normally.

LOW: balance = 1-3 days remaining
 Dashboard: "Signal credits critically low."
 Telegram/email alert to user.
 System reduces API call frequency (save credits):
 - Strategy refresh: 2h → 4h
 - Non-critical calls skipped

DEPLETED: balance = 0
 Dashboard: "Signal credits depleted. Running on
 basic signals."
 Mode → BASIC
 Existing positions managed normally.
 New entries use local basic signals.

TOPPED UP: user adds x402 balance
 Mode → FULL (automatic, no restart needed)
 "Full intelligence restored."

Estimated Usage Tracking

```
class X402Monitor:
    def __init__(self):
        self.daily_cost_history = [] # last 7 days of costs
```

```

def estimated_days_remaining(self, balance):
    if not self.daily_cost_history:
        return float('inf') # no data yet
    avg_daily = sum(self.daily_cost_history) / len(self.daily_cost_history)
    if avg_daily == 0:
        return float('inf')
    return balance / avg_daily

def check_balance(self, balance):
    days = self.estimated_days_remaining(balance)
    if days > 7:
        return "healthy"
    elif days > 3:
        return "warning"
    elif days > 1:
        return "low"
    else:
        return "depleted"

```

PROTECTION LAYER 5: POSITION SAFETY ON ANY FAILURE

Regardless of signal mode, ALL positions are protected at ALL times.

INVARIANTS (never violated, regardless of signal availability):

1. Every position has an on-chain stop on HL
 Stops are placed on HL's SERVER, not dependent on ZERO OS running.
 If ZERO OS crashes, HL still executes the stop.
2. Immune system runs independently of signals
 Stop verification happens every 60 seconds.
 Position sync happens every 60 seconds.
 Equity monitoring happens every 60 seconds.
 NONE of these require the signal API.
3. No position is ever left "naked"
 If a stop is missing → emergency close (market order)
 If desync detected → reconcile from HL
 These protections work in ALL modes (FULL, CACHED, BASIC, PROTECTION)
4. Protection mode winds down safely
 Existing positions close via their normal exit rules:
 - Stop loss triggers → position closes

- Max hold time reached → position closes
 - Signal reversal (if cached signals available) → position closes
- After all positions close, system enters idle state.
No new positions opened. Equity preserved.

5. User can ALWAYS force-close everything
zeroos emergency-close
→ Closes all positions immediately (reduce_only market orders)
→ Cancels all open orders
→ Regardless of signal mode or API status

PROTECTION LAYER 6: API PROVIDER ABSTRACTION

The signal API is accessed through an interface, not directly. If the provider changes, ZERO OS swaps the adapter.

```
class SignalProvider:
    """Abstract interface. Any signal source implements this."""

    def check_signals(self, coin, expressions) -> SignalResult:
        raise NotImplementedError

    def assemble_strategy(self, coin) -> StrategyResult:
        raise NotImplementedError

    def optimize_portfolio(self, coins) -> PortfolioResult:
        raise NotImplementedError

class NVProtocolProvider(SignalProvider):
    """Current provider. Calls arena.nvprotocol.com via x402."""

    def check_signals(self, coin, expressions):
        response = self.api_call('/signals/check', data, x402=True)
        self.cache.save(coin, response)
        return response

class BasicProvider(SignalProvider):
    """Fallback. Computes RSI/EMA/MACD locally from HL price data."""

    def check_signals(self, coin, expressions):
        prices = self.hl_websocket.get_prices(coin)
        return self.compute_basic_indicators(prices)

class CachedProvider(SignalProvider):
```

```

        """Uses last known API data from local cache."""

    def check_signals(self, coin, expressions):
        cached = self.cache.load(coin)
        if cached and not cached.is_expired():
            return cached
        raise CacheExpiredError()

# The evaluator uses the provider interface:
class Evaluator:
    def __init__(self, provider: SignalProvider):
        self.provider = provider

    def evaluate(self, coin):
        signals = self.provider.check_signals(coin, self.expressions)
        # Rest of evaluation logic is IDENTICAL regardless of provider

```

Why This Protects Your IP

The SignalProvider interface means:

1. ZERO OS works with ANY signal source, not just one
2. If the current provider disappears, swap to another
3. The OS layer (everything below the provider) is 100% yours
4. A future signal provider can be integrated in hours, not months
5. You could even build your own signal engine eventually – just implement the SignalProvider interface

The interface IS the IP protection.

It decouples your value from any single dependency.

WHAT ZERO OS OWNS (Your IP)

YOURS (survives loss of any external dependency):

- └─ Immune system (13 checks, self-audit, desync detection)
- └─ Multi-agent orchestrator
- └─ Claude reasoning engine
- └─ Basic signal engine (RSI, EMA, MACD – local, free)
- └─ Signal cache layer
- └─ Graceful degradation system (4 modes)
- └─ x402 balance monitoring
- └─ Position safety invariants
- └─ SignalProvider abstraction interface

- └─ CLI (zeroos init/start/status)
- └─ Encrypted keystore
- └─ Dashboard telemetry
- └─ getzero.dev (frontend, design, brand)
- └─ Paper trading executor
- └─ Terminal aesthetic and UX design
- └─ All 50+ spec documents from this session

NOT YOURS (external dependency):

- └─ Signal API backend (NVProtocol/arena.nvprotocol.com)
- └─ Signal scoring algorithms (server-side)
- └─ Strategy assembly tournament (server-side)
- └─ Portfolio optimization (server-side)
- └─ 85M+ historical indicator data (server-side)

The external dependency makes ZERO OS BETTER. But ZERO OS exists, runs, protects positions, and provides basic trading capability WITHOUT it. That's the self-protection guarantee.

IMPLEMENTATION

 Session: Self-Protection Layer (2 hours)

1. Implement SignalProvider interface in Python:
 - SignalProvider (abstract base)
 - NVProtocolProvider (current API, x402)
 - CachedProvider (local cache)
 - BasicProvider (RSI/EMA/MACD from HL prices)
2. Implement SignalManager with 4 modes:
 - FULL → CACHED → BASIC → PROTECTION
 - Automatic fallback on API failure / payment depletion
3. Implement local cache:
 - ~/.zeroos/cache/ directory
 - Save every API response to disk
 - Load on startup before API call
 - Cache age tracking + freshness rules
4. Implement x402 balance monitor:
 - Track daily cost, estimate days remaining
 - Alert at WARNING/LOW/DEPLETED thresholds
5. Implement basic signal engine:

RSI(14) + EMA(9,21) + MACD from HL WebSocket prices
Basic signal scoring (direction + confidence, no Sharpe)
Good enough for protection mode, not for alpha generation

6. Add zeroos emergency-close command:

Force-close all positions immediately
Works regardless of signal mode or API status

7. Update evaluator to use SignalProvider interface:

Evaluator doesn't know or care where signals come from
Same gates, same logic, different data quality

Test: disconnect from API, verify system degrades to CACHED,
then to BASIC, then to PROTECTION. Verify no positions are
left unprotected at any mode transition.