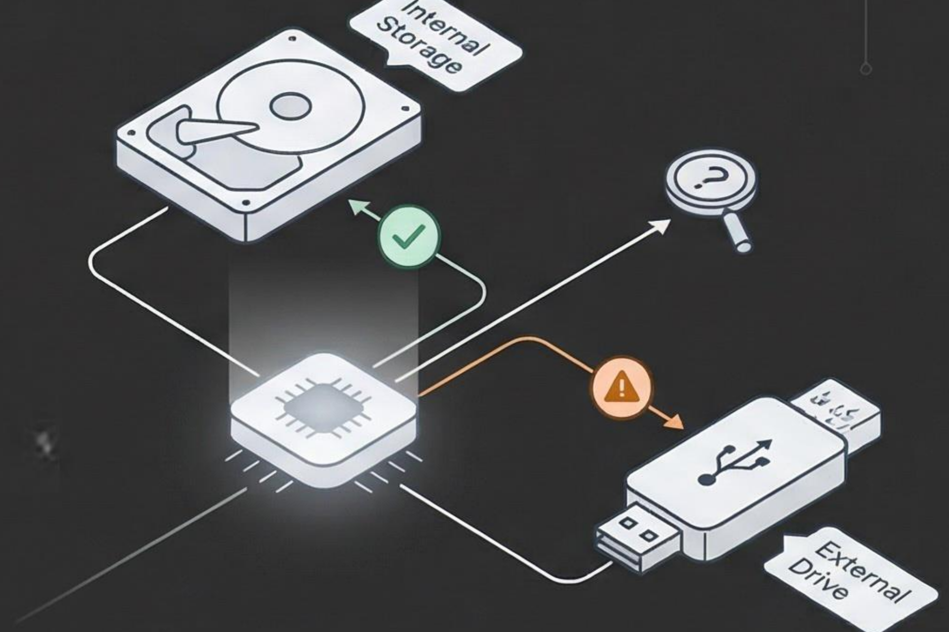
Unified Storage Health Monitoring on macOS

SMART-Aware & SMART-Fallback Architecture

A disk health monitoring system that adapts when SMART data is unavailable



The Problem

Disk health monitoring relies heavily on SMART

On macOS:

- External USB drives often do not expose SMART
- Monitoring tools fail or return incomplete data

Impact

No reliable health signal for a large class of storage devices

Design Goals

Design Objectives

1. Detect all physical disks
2. Identify SMART capability per disk
3. Provide a health score for every disk
4. Avoid false confidence or silent failures

Core Principle

Graceful degradation instead of hard dependency on SMART

High-Level Architecture

System Architecture

Pipeline-based design

Each stage is modular and replaceable

Disk Detection

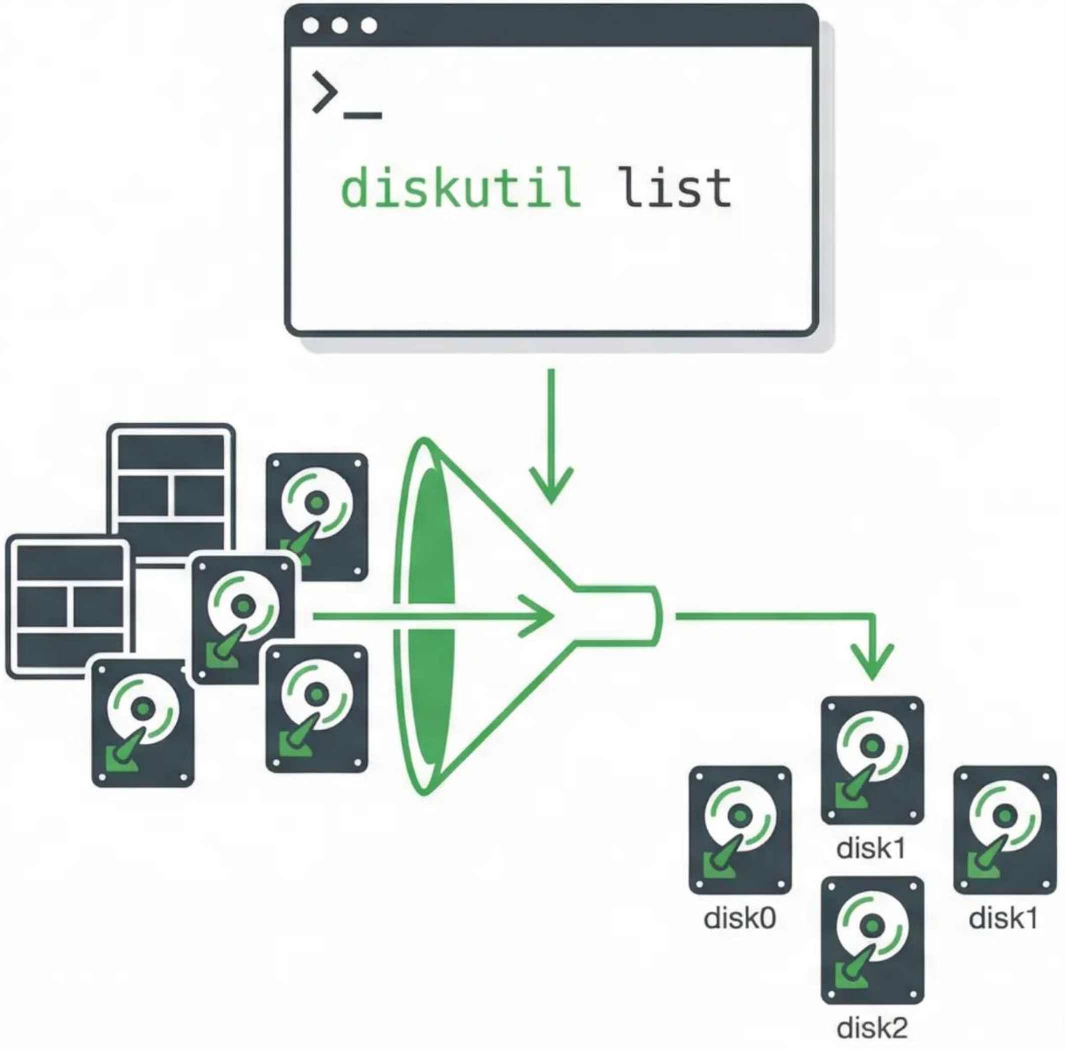
Step 1: Disk Detection

Action

- Run diskutil list
- Filter physical disks only

Why

- Logical volumes can hide underlying hardware behavior
- Health must be evaluated at the physical layer



SMART Capability Detection

Step 2: SMART Verification

Action

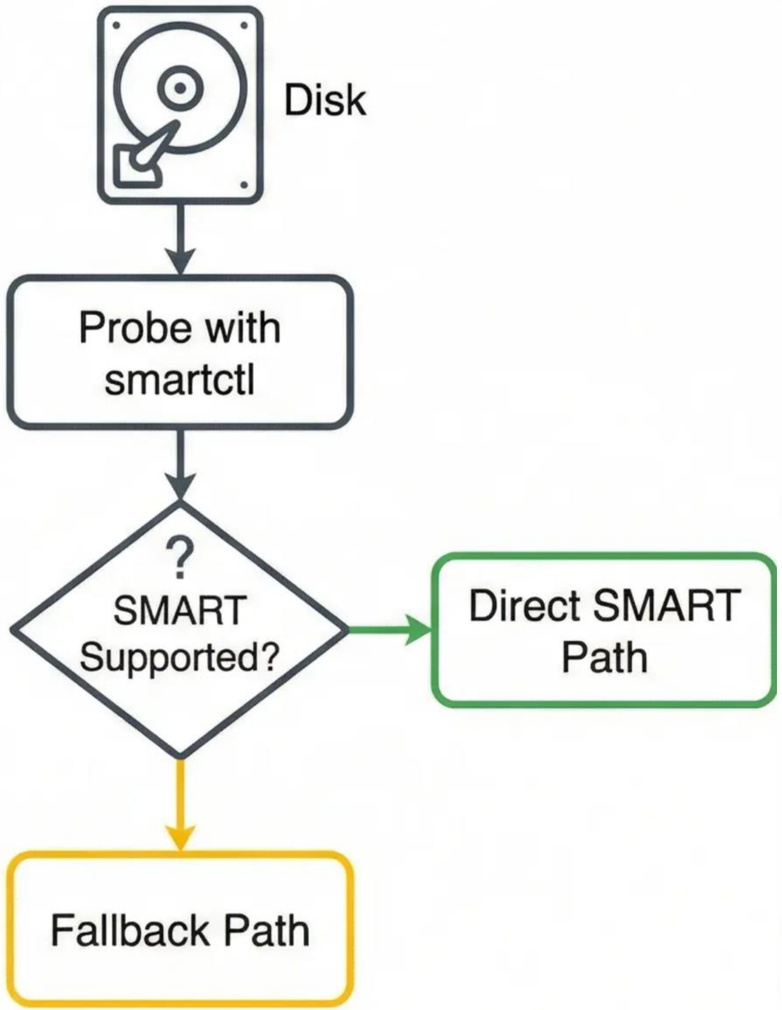
Probe each disk using smartctl

Decision Gate

- SMART supported → Direct SMART path
- SMART not supported → Fallback path

Insight

SMART availability ≠ disk health



Direct SMART Path

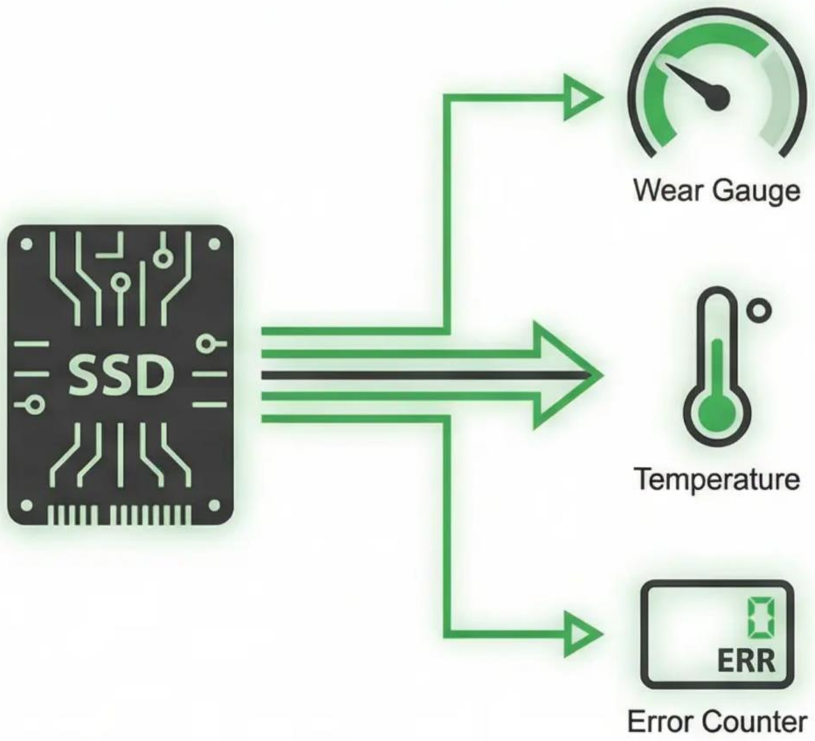
Step 3A: SMART-Based Analysis

Used when SMART is supported

Parse NVMe / ATA SMART attributes

Evaluate:

- Wear level
- Temperature
- Media & data integrity errors



Output

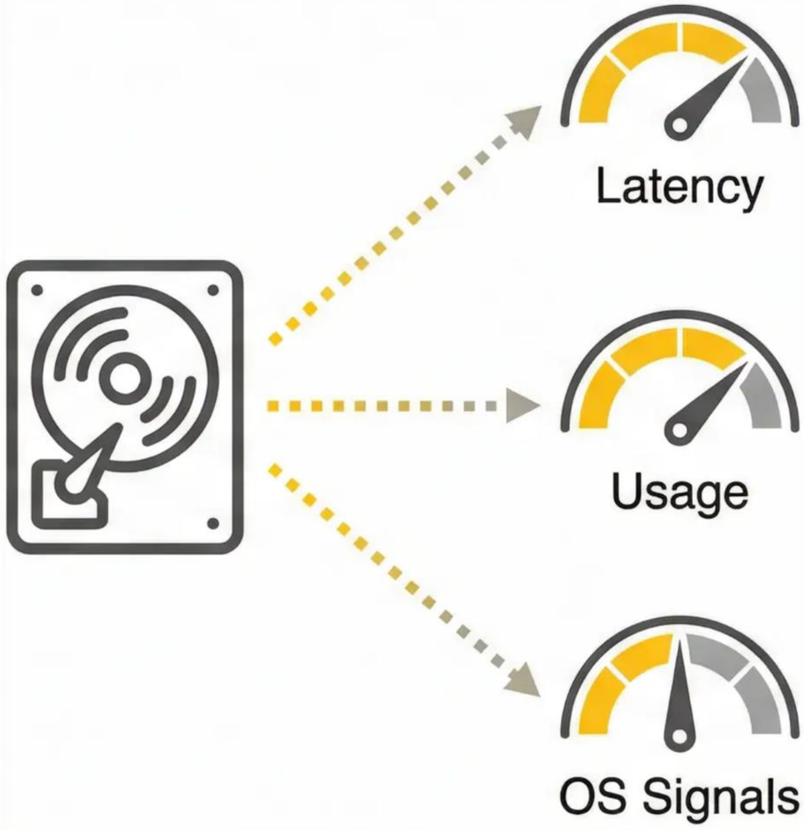
High-fidelity hardware health signal

Fallback Path (No SMART)

Step 3B: Indirect Health Analysis
Used when SMART is unavailable

Measure:

- I/O latency
- Disk usage pressure
- OS-level indicators



Philosophy

If internals are hidden, observe external behavior

Health Scoring & Normalization

Step 4: Health Processing

Convert raw signals into a 0–100 health score

Apply consistent thresholds:

HEALTHY | **WARNING** | **CRITICAL**

Benefit

Comparable health signals across heterogeneous disks

Unified Reporting

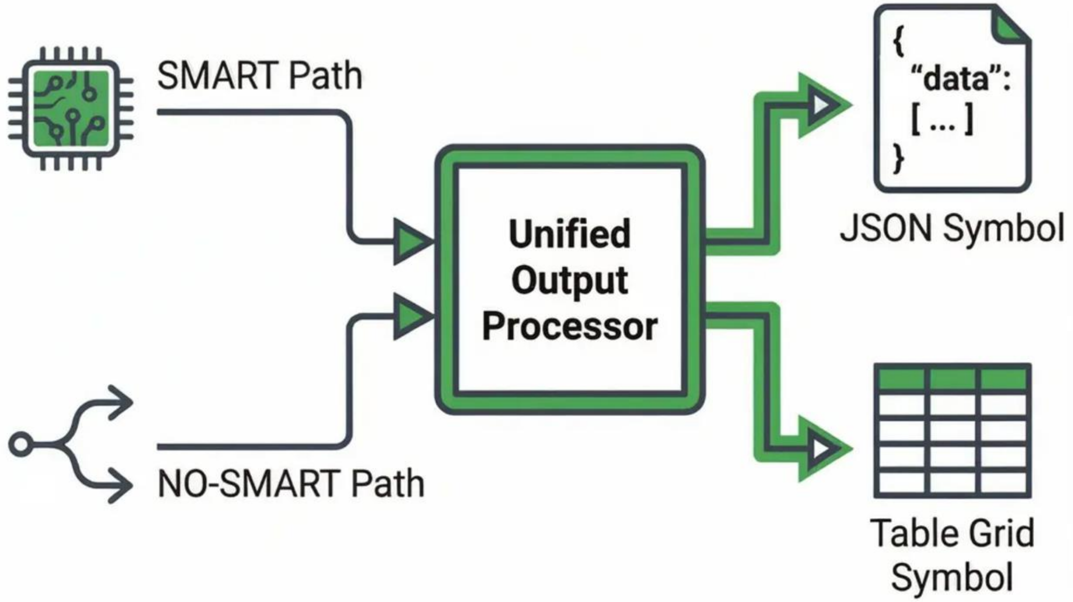
Step 5: Final Output

Consolidation

- Merge SMART and non-SMART results

Formats

- JSON → automation & integrations
- CLI table → human readability



Single source of truth

Example Output

Sample Result

Disk	Mode	Score	Status
/dev/disk0	SMART	98	HEALTHY
/dev/disk4	NO_SMART	100	HEALTHY

Key Point

Every disk is evaluated, none are ignored

Known Limitations

Known Constraints

- USB bridges may block SMART on macOS
- Indirect health is inferential, not diagnostic
- Vendor-specific SMART attributes vary
- Read-only health monitoring
- macOS-specific tooling

Limitations are explicit, not hidden

Engineering Trade-offs

Design Trade-offs

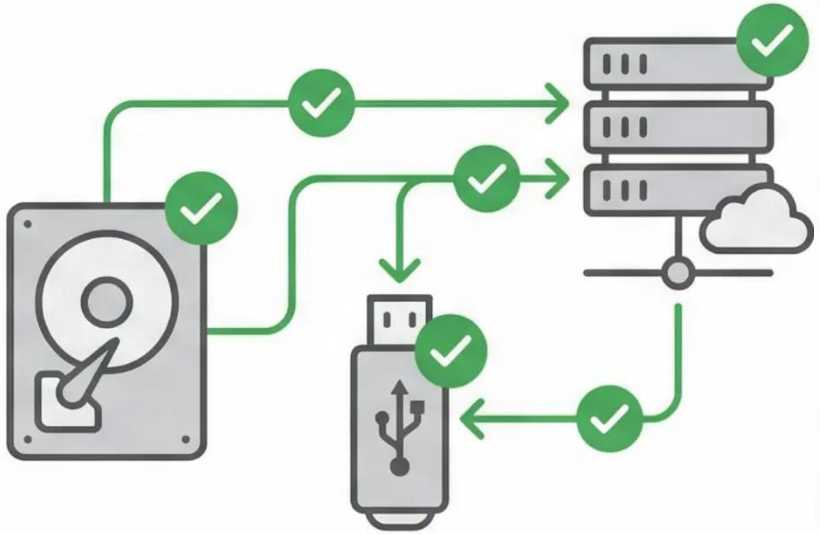
Option	Decision
Perfect telemetry	❌ Not guaranteed
Partial insight	✅ Always available
Fail on missing SMART	❌
Adaptive strategy	✅

Optimized for reliability, not perfection

Engineering Value

Why This Matters

- Reflects real-world infrastructure constraints
- Prevents blind spots in monitoring
- Scales across internal & external storage



Mindset

Observability-first engineering

Interview Takeaway

Final Message

"I designed a storage health system that treats SMART as an optimization, not a dependency, ensuring reliable health signals even when hardware telemetry is partially unavailable."

Observability-first engineering approach