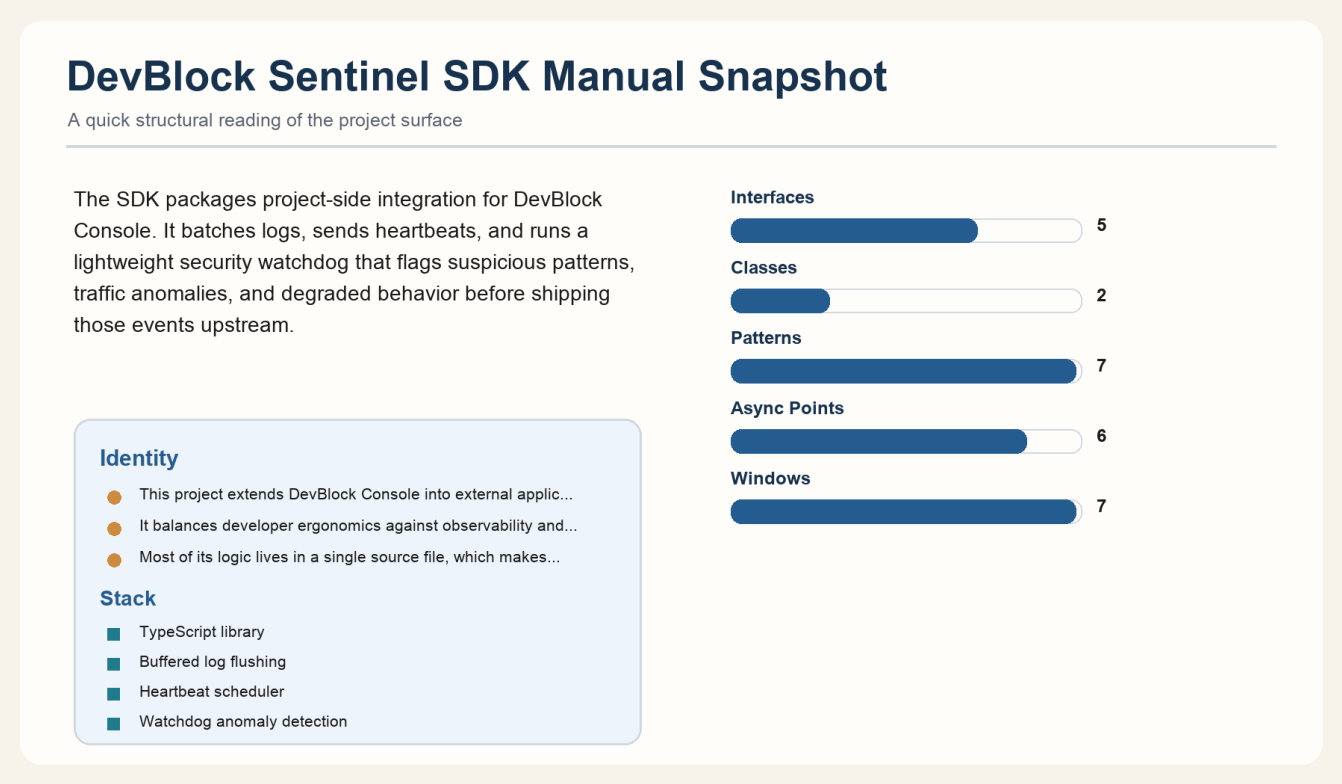
DevBlock Sentinel SDK Manual

Telemetry, batching, heartbeat, and security-watchdog behavior

The SDK packages project-side integration for DevBlock Console. It batches logs, sends heartbeats, and runs a lightweight security watchdog that flags suspicious patterns, traffic anomalies, and degraded behavior before shipping those events upstream.



DevBlock Sentinel SDK Manual visual snapshot

What this project is for

The SDK packages project-side integration for DevBlock Console. It batches logs, sends heartbeats, and runs a lightweight security watchdog that flags suspicious patterns, traffic anomalies, and degraded behavior before shipping those events upstream.

> **Primary audience: Use this when integrating customer projects into DevBlock Console or changing the telemetry, buffering, and watchdog contract.**

How to identify it quickly

- This project extends DevBlock Console into external applications rather than rendering product UI.
- It balances developer ergonomics against observability and false-positive risk.
- Most of its logic lives in a single source file, which makes reading easy and maintenance dense.
- Its behavior affects perceived trust in the broader platform because it determines what gets reported upstream.

Technology stack

- TypeScript library
- Buffered log flushing
- Heartbeat scheduler
- Watchdog anomaly detection
- Circuit breaker and retry logic

Structural counts

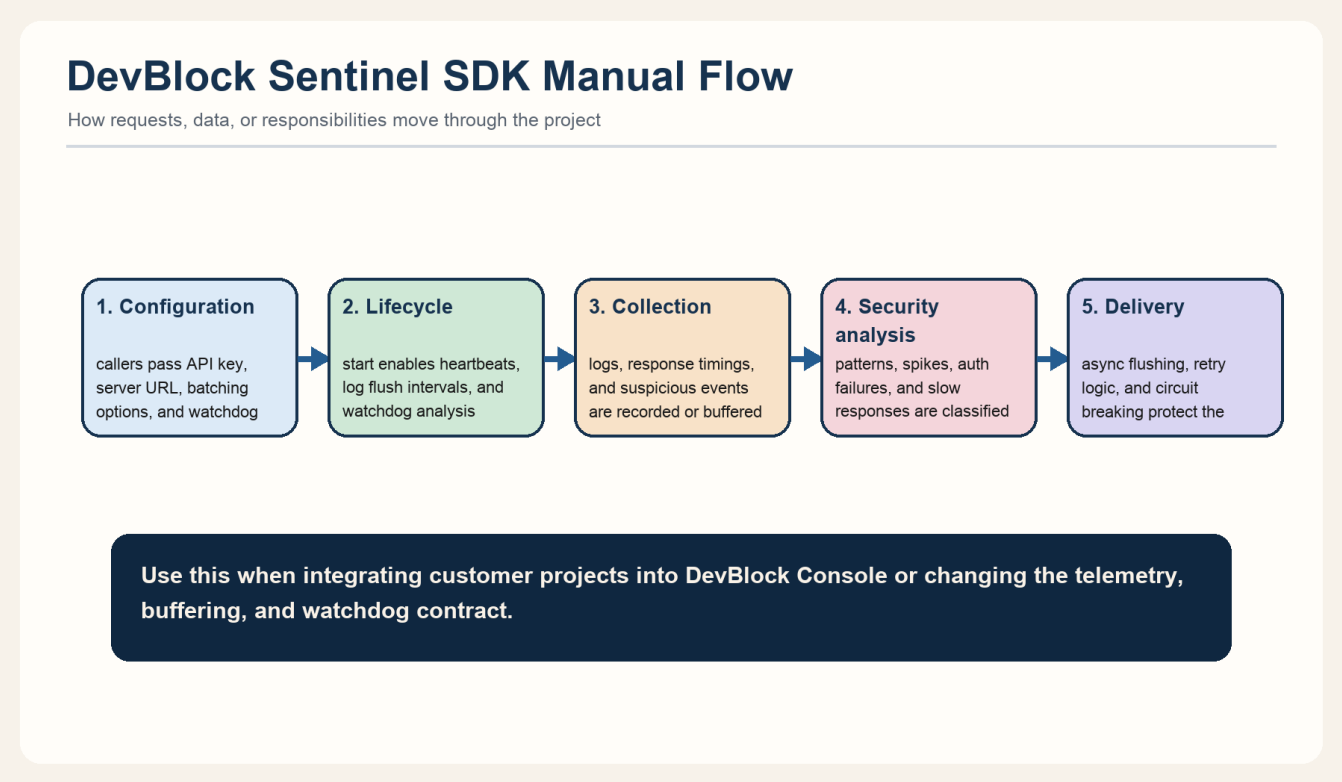
Metric	Count
Interfaces	5
Classes	2
Patterns	7
Async Points	6
Windows	7

Architecture reading

- The SDK acts like a small runtime engine rather than a thin HTTP wrapper.
- Watchdog defaults are part of the product contract because they shape alert quality and noise.
- Sliding windows are core to the anomaly model, so threshold edits are architectural changes, not cosmetic tweaks.
- Circuit-breaker behavior protects both the SDK host and the backend platform under failure.

Core flow

This sequence is the shortest useful mental model for how the project behaves at runtime or during operator use.



DevBlock Sentinel SDK Manual responsibility or data flow

Responsibility matrix

DevBlock Sentinel SDK Manual Responsibility Matrix

What this project owns versus what it coordinates

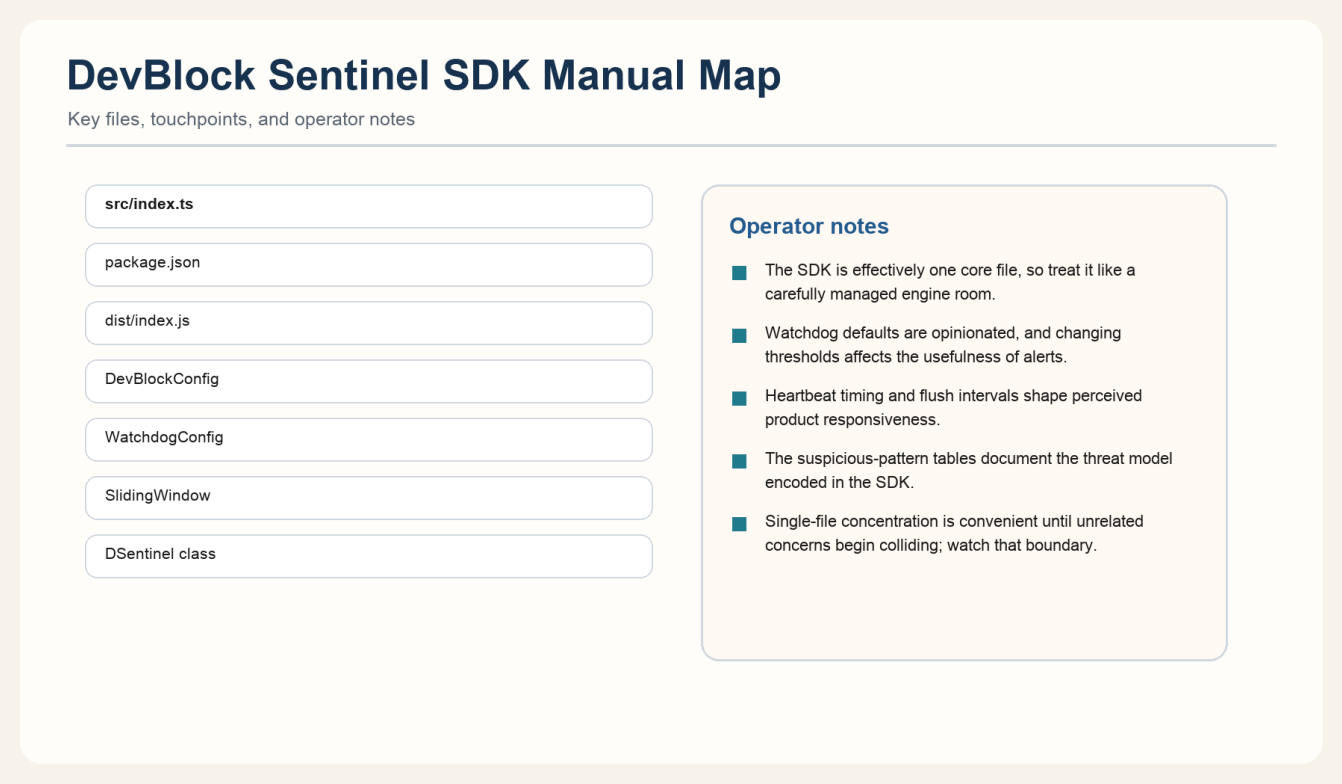
Area	Owns	Coordinates with	Failure mode
Product role	This project extends DevBlock Console into external applications rather than rendering product UI.	It balances developer ergonomics against observability and false-positive risk.	A single-file implementation makes local reasoning easy but can become dense fast.
Main inputs	Configuration: callers pass API key, server URL, batching options, and watchdog overrides	Lifecycle: start enables heartbeats, log flush intervals, and watchdog analysis timers	False positives in watchdog heuristics can damage trust if thresholds drift.
Change surface	Read the public config interfaces first because they define how host projects experience the SDK.	Trace `start`, buffering, and watchdog analysis together before changing intervals or flush	Any transport failure policy change affects both telemetry completeness and backend load.
Operator posture	Validate behavior with scenario-based reasoning, not only compile success.	Threshold changes should be reviewed against realistic traffic patterns to avoid noisy alerts.	The SDK is effectively one core file, so treat it like a carefully managed engine room.

DevBlock Sentinel SDK Manual ownership matrix

Key files to read first

If you are new to this project, read these files in order before making broad edits.

- src/index.ts
- package.json
- dist/index.js
- DevBlockConfig
- WatchdogConfig
- SlidingWindow
- DSentinel class



DevBlock Sentinel SDK Manual orientation map

Change workflow

- Read the public config interfaces first because they define how host projects experience the SDK.
- Trace `start`, buffering, and watchdog analysis together before changing intervals or flush semantics.
- Treat suspicious-pattern lists as encoded threat-model decisions rather than random regex collections.
- Any change to transport or retry policy should be reviewed in terms of both data loss and backend pressure.

Operator notes

- The SDK is effectively one core file, so treat it like a carefully managed engine room.
- Watchdog defaults are opinionated, and changing thresholds affects the usefulness of alerts.
- Heartbeat timing and flush intervals shape perceived product responsiveness.
- The suspicious-pattern tables document the threat model encoded in the SDK.
- Single-file concentration is convenient until unrelated concerns begin colliding; watch that boundary.

Validation posture

- Validate behavior with scenario-based reasoning, not only compile success.
- Threshold changes should be reviewed against realistic traffic patterns to avoid noisy alerts.
- Changes to buffering or stop behavior should consider shutdown and crash boundaries explicitly.

Current risks or watchpoints

- A single-file implementation makes local reasoning easy but can become dense fast.
- False positives in watchdog heuristics can damage trust if thresholds drift.
- Any transport failure policy change affects both telemetry completeness and backend load.

Glossary

Term	Meaning
Heartbeat	A periodic signal used to show that a host application is still alive.
Sliding window	A rolling time-based counter used for rate and anomaly analysis.
Circuit breaker	A guard that temporarily stops outbound attempts after repeated failure.