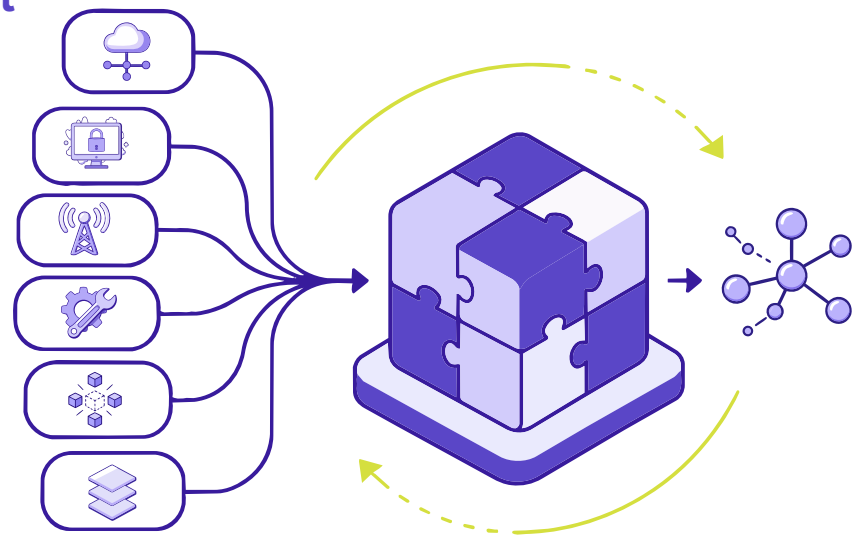
INSIDE THE SDK.

From Config file to any asset type integration

A look at the six components that turn a declarative manifest into a deployed, monitored, and continuously patchable Go microservice feeding the WanAware Relationship Graph, no-code, AI-augmented, and identical whether the asset, IT, IoT, OT, AI, cloud, app, service, or a proprietary system nobody else ever built a connector for.



THE SIX COMPONENTS OF THE PLATFORM

Every integrator the SDK produces is assembled from the same six subsystems, whether the provider is a hyperscale cloud, an EDR platform, or a piece of enterprise networking hardware. Each subsystem has one job, and together they turn a configuration file into a production-grade, continuously monitored service.

Component	What it does
1. Code Generation Engine (gen)	Compiles asset provider configuration into a deployable Go microservice: WebSocket client, chunked cloud storage upload pipeline, rate-limited worker pools, and embedded OpenTelemetry tracing.
2. Asset Categorization & Tier-0 Engine	Translates vendor-specific metadata into the platform catalog via mapping, match and survivorship configurations, compiled into the binary and auto-registered at startup.
3. Local Validation & QA Gates	Offline commands (validate, status, shipcheck) that catch illegal subtypes, unmapped fields, and unconfirmed AI proposals before anything reaches the platform.

Component	What it does
4. AI-Agent Orchestration & MCP Server	Exposes catalog, validate, status, gen, patch, and shipcheck as structured tools to any MCP-compliant coding agent, plus Claude Code slash commands.
5. Accuracy Measurement & Reference Evaluation	Scores AI-suggested mappings against a human-validated reference set across subtype precision, role precision, catalog integrity, and run-to-run determinism.
6. Operational Lifecycle & Telemetry	Containerized deployment, bi-directional WebSocket command channel, and continuous OpenTelemetry health, crawl, and error-rate reporting.

ONE ARCHITECTURE, EVERY ASSET CATEGORY

The six components above do not change shape depending on what is being connected. A hyperscale cloud account, an edge gateway, a SCADA historian, and a homegrown asset tracker all pass through the identical generation, categorization, and quality-gate path. What differs between asset categories is never the architecture. It is only the handful of lines in sdk-config.yaml that describe how that specific thing exposes its data.

Asset category	What is specific to it	What stays identical
IT and cloud	Hyperscale cloud and SaaS authentication flows, ITSM and ticketing API shapes.	Same gen pipeline, same Tier-0 catalog, same four-stage gate.
IoT and the network edge	Lightweight device and gateway APIs, often high device counts per integrator.	Same event-stream wiring into the Relationship Graph.
AI systems and services	Model registries and LLM endpoint schemas, prompt and context telemetry.	Same validation gates, same catalog registration model.
OT and industrial	Safety-Critical Segment handling, passive-only discovery for fragile equipment.	Same worker-pool concurrency and schema-registration model.
Proprietary and niche platforms	A custom or undocumented export format, described once in config.	Same two-tier file ownership and safe regeneration guarantees.

That uniformity is what makes the SDK's reach a function of configuration effort rather than engineering specialization. Nobody needs a different skill set to connect an industrial historian than to connect a hyperscale cloud account, because the platform-side half of that connection was never different to begin with.

THE FOUR-STAGE QUALITY GATE PIPELINE

Every integrator, human-authored or AI-generated, has to clear the same verify pipeline before it is considered shippable. The four stages run sequentially, and AI agent execution is held to strict iteration caps so a resolving error cannot spiral into an infinite loop.

Stage	What runs	What it guarantees
1. Compile	Go toolchain (go build).	Zero syntax errors, valid types, complete dependency resolution.
2. Contract tests	Native test suite (go test).	Provider response parsers conform exactly to platform ingestion interfaces.
3. Response replay	Replay engine (internal/replay).	Resilience against real-world payload variation, verified against recorded provider responses.
4. Shipcheck	integrator-sdk shipcheck.	go vet, gofmt, unit tests, catalog validation, and idempotent regeneration, all in one gate.

AI CONTEXT THAT IS GROUNDED, NOT GENERIC

Agents working inside the SDK are not relying on general-purpose training weights to guess at a provider's API behavior. The codegen/ecosystem embeds curated, version-pinned research notes on API idiosyncrasies, rate-limit envelopes, and authentication flows for each supported provider, alongside playbooks that dictate exactly how an agent constructs discovery loops and translates nested structures into platform definitions. These are pinned to the running SDK release, so an agent is always working against the exact API surface the platform actually supports.

WHAT THE PIPELINE CATCHES BEFORE IT SHIPS

A generated AWS integrator compiles cleanly and passes contract tests, but the response replay stage flags that a recently observed API payload includes a nested field the parser does not expect. Shipcheck never runs. The build fails in CI, not in a customer's environment.

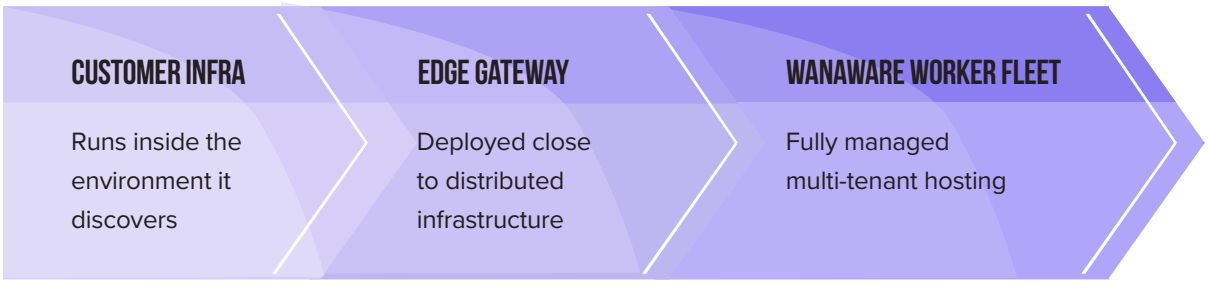
Where human review sits in the loop

AI-assisted modification is confined to designated provider directories and cannot touch platform core modules. Every AI-proposed mapping is written as confirmed: false, and the shipcheck gate will not pass a configuration file that still contains an unconfirmed proposal. A human engineer has to explicitly review and elevate it first, and any field an agent leaves unmapped has to carry an explicit justification code from a fixed, platform-enforced list.

HOW AN INTEGRATOR ACTUALLY RUNS

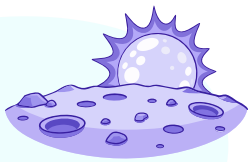
A compiled integrator deploys as a lightweight, containerized Go microservice, on customer-managed infrastructure, an edge gateway, or inside the multi-tenant WanAware Platform Worker Fleet. Once running, it communicates bi-directionally over a persistent WebSocket connection: scheduled polling, ad-hoc administrator crawls, and configuration refreshes flow in, while telemetry, crawl metrics, and error rates flow out continuously to WanAware's operational dashboards.

- **Read-only by design.** Integrators use credential-delegated, read-only API access, such as an AWS IAM AssumeRole configuration, and never install an agent on the target node itself.
- **99.9% uptime commitment.** Applies to platform ingestion endpoints and core worker fleets, with near real-time propagation of configuration changes.
- **Scoped discovery.** Runtime filters enforce selective resource discovery, so an integrator only pulls the data it was configured to pull, not an entire provider account by default.
- **Safety-Critical Segment handling for fragile equipment.** For OT and industrial assets, the same deployment model automatically restricts discovery to passive telemetry wherever active probing would put a PLC, SCADA system, or other fragile controller at risk.



ONE ARCHITECTURE, EVERY PROVIDER

The Integrator SDK empowers organizations to accelerate integration velocity and achieve predictable, enterprise-grade reliability across all asset types. By leveraging a single uniform architecture, teams eliminate technical debt and specialized engineering overhead while ensuring zero downtime, strict quality control, and effortless scalability from cloud to edge.

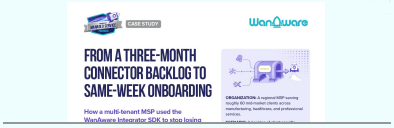


NEW HORIZONS

The architecture does not get rewritten for the next provider. Only the config file does.

See how the Integrator SDK turns configuration into a flexible, ready-to-use integration.

Read the Case Study Here





READY FOR LIFTOFF?

See every campus, every device, and every gap in one graph.

FREE TRIAL

SEE A DEMO