INTEGRATOR SDK
INDUSTRY

TECHNICAL WHITEPAPER

WanAware

FROM MONTHS OF BESPOKE CODE TO A NO-CODE, AI-AUGMENTED ASSET INTEGRATOR



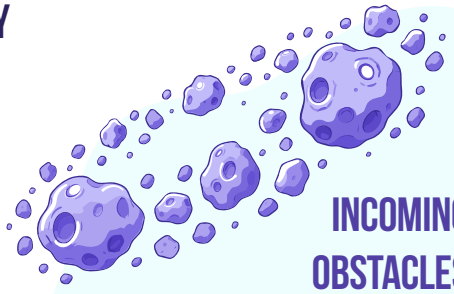
Every disconnected tool and asset type in your estate is another connector someone has to hand-build and hand-maintain. This white paper describes how WanAware accelerates any asset, IT, IoT, OT, AI, cloud, app, service, or proprietary, into a working integration to the WanAware platform in hours, no code required, with the guardrails to trust what it produces.

TABLE OF CONTENTS

THE INTEGRATION CRISIS UNDERNEATH EVERY ASSET INVENTORY	3
FROM MANUAL CODING TO DECLARATIVE GENERATION	3
Alert fatigue and manual triage	4
Safe regeneration, not a one-time scaffold	5
ANY ASSET TYPE, ACCELERATED INTO WANAWARE	5
WHAT THE SDK ACTUALLY SOLVES	6
WHAT CHANGES, IN NUMBERS	7
THE FOUR-STAGE QUALITY GATE PIPELINE	8
GOVERNANCE BUILT FOR AI-ACCELERATED ENGINEERING	8
Six tools, one MCP server	9
MEASURING WHETHER THE AI MAPPING CAN BE TRUSTED	9
WHO THIS CHANGES THINGS FOR	10
BUILT TRANSPARENTLY: CURRENT PLATFORM BOUNDARIES	11
BUILT FOR CUSTOMERS AND THE PARTNERS WHO SERVE THEM	11
CONCLUSION: INTEGRATION AS A DELIVERABLE, NOT A LIABILITY	12

THE INTEGRATION CRISIS UNDERNEATH EVERY ASSET INVENTORY

Modern enterprise infrastructure has expanded past the data center into distributed multi-cloud platforms, ephemeral containers, software-defined networks, and specialized security tooling. Most organizations run dozens of disconnected monitoring, security, cloud, and networking tools, each holding a partial, isolated view of the estate. Unifying them into a single asset intelligence platform has historically meant hand-building a bespoke connector for every endpoint API: custom pagination, custom retry logic, custom authentication, custom data formatting, repeated for every vendor, forever.



INCOMING OBSTACLES

Engineering capacity spent on pagination loops and retry logic is engineering capacity not spent on the thing you are actually building.

The cost of that approach shows up everywhere. Fewer than 40 percent of organizations trust their current asset data. More than half report that an inaccurate inventory directly triggered an outage, a security vulnerability, or a performance failure in the past year. Up to 25 percent of infrastructure spend is wasted on untracked compute instances, unattached storage, and abandoned IP allocations. And for MSPs, GSIs, and ISVs, every one of those connectors is 80 to 120 engineering hours of plumbing that has nothing to do with the capability they are actually trying to sell.

MSPs, GSIs, and ISVs typically burn 80 to 120 billable engineering hours integrating a single new provider under a traditional build. The Integrator SDK's gen command produces the baseline service in about 60 minutes, an 80 percent or greater reduction in development overhead.

FROM MANUAL CODING TO DECLARATIVE GENERATION

The **WanAware Integrator SDK** turns integration development from a manual software engineering effort into a deterministic, configuration-driven, AI-accelerated process. A developer, or an AI coding agent, describes an asset provider's endpoints, resources, and extraction behavior in a single manifest.. The SDK compiles that declarative specification into a deployable Go microservice, implementing in code what every hand-built connector has historically had to reinvent from scratch.

The architectural premise is strict separation of operational boilerplate from provider-specific logic. The SDK generates the transport plumbing, concurrency management, and schema registration. The developer or agent is responsible only for the targeted authentication and data-extraction routines that are actually specific to that vendor.

Five patterns every hand-built connector reinvents

Bespoke integration engineering suffers from architectural degradation over time: hand-rolled backoff logic, inconsistent identifier derivation, and worker pools that mishandle memory under high API volume all introduce their own unpredictable failure modes. The generated codebase automatically implements five patterns that remove the need to solve any of that by hand again.

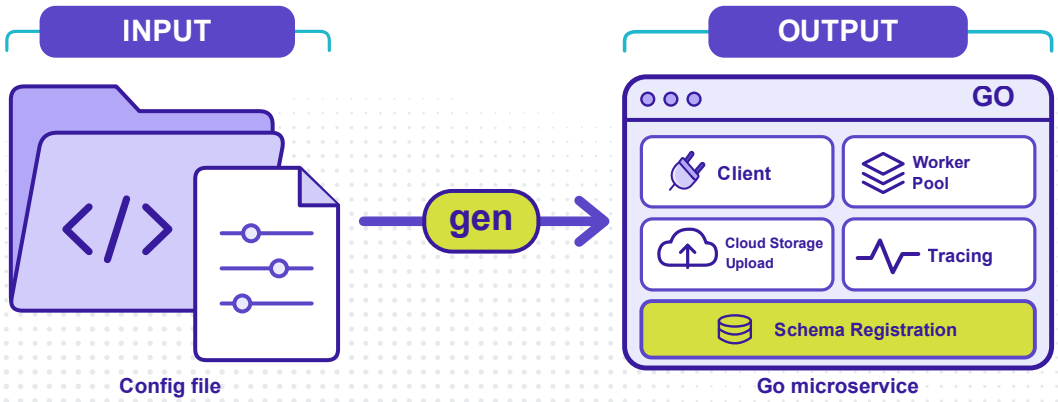
Pagination that does not break.
Standardized cursor and offset pagination loops systematically traverse provider API responses without buffer overflows or infinite loops, the single most common place a hand-built connector quietly fails.

Deletions that do not crash the pipeline.
Deterministic intercepts for missing or deleted upstream objects prevent integration crashes and trigger the appropriate tombstone lifecycle within the asset graph, instead of leaving a stale record behind.

Identifiers that are globally unique by construction.
Uniform string parsing algorithms sanitize provider-native identifiers into globally unique platform external IDs, removing the inconsistent identifier logic that is the most common root cause of duplicate asset records.

Discovery that stays in scope.
Dynamic runtime filters enforce selective resource discovery, preventing unwanted API consumption or out-of-scope asset gathering, so an integrator only pulls what it was actually configured to pull.

Change events that reach the platform in real time.
Decoupled notification structures transmit real-time change events directly to WanAware event streams, instead of waiting for the next scheduled crawl to notice something changed.



Safe regeneration, not a one-time scaffold

Most code generators create a static template once; any later re-run risksoverwriting manual work. The Integrator SDK enforces a strict two-tier file ownership model instead. Infrastructure files, scaffolding, configuration readers, connection managers, storage upload modules, telemetry bindings, are SDK-owned and safely overwritten on every run, so upstream security patches and performance improvements propagate instantly across the entire connector catalog. Provider-specific authentication handlers and payload parsing logic are isolated into stub files created once and never touched again. Customers get an integration fleet that can be patched without breaking custom logic. Partners get to continuously update their commercial connectors against new SDK releases without risking regressions in the intellectual property they built on top of it.

WHAT GENERATION LOOKS LIKE IN PRACTICE

A developer describes a provider in a document, no hand-written transport code, no manual schema authoring, and runs the SDK. In validation testing against the AWS provider catalog, that no-code, AI-augmented process took autonomous agents from a raw service specification to a fully compiling, contract-tested Go service in about 60 minutes, with zero hand-written boilerplate.

ANY ASSET TYPE, ACCELERATED INTO WANAWARE

The real limit on how much of an environment ever made it into an asset intelligence platform was never curiosity. It was engineering bandwidth. Hand-building a connector required someone who understood both the target platform's API and WanAware's own ingestion model, the catalog, the **Relationship Graph**, the event streams, well enough to wire the two together correctly. That combination of expertise is scarce, so historically only a handful of platforms with enough customer demand to justify months of engineering ever got built.

The SDK removes that gate. Because generation produces WanAware's entire half of the connection automatically, transport, concurrency, schema registration, event streaming, the only expertise anyone has to bring is knowledge of the thing in front of them: how a given platform or device actually exposes its data. Nobody has to learn WanAware's internals to connect something to it, and no one has to write code to do it.

- **IT and cloud.**
  Hyperscale clouds, SaaS platforms, container orchestration, and ITSM or ticketing systems follow the identical config-to-service path as anything else WanAware connects to.

- **IoT and the network edge.**
  Connected sensors and edge gateways resolve into the same Relationship Graph through the same generation process, with no separate integration methodology to learn for a different asset category.

- **AI systems and services.**
  Machine learning models, inference endpoints, LLM applications, and autonomous agents map seamlessly into the Relationship Graph alongside traditional assets, maintaining unified governance across disparate AI providers.

- **OT and industrial.**
  PLCs, SCADA systems, and industrial controllers connect through the same SDK, with Safety-Critical Segment handling applied automatically wherever active discovery would put fragile equipment at risk.

- **Proprietary and niche platforms.**
  A regional OSS system, a homegrown asset tracker, a vendor-discontinued legacy tool, the systems that never justified a bespoke, multi-month build now take an afternoon with a config file instead.

That is the real shift. Acceleration is not just about building a connector faster. It is about removing the gate that decided, by default, which asset types were even worth connecting in the first place.

WHAT ACCELERATION LOOKS LIKE FOR A NICHE ASSET

A utility running a 15-year-old, vendor-abandoned SCADA historian describes its proprietary export format in a single config file. The SDK generates the service, applies the Safety-Critical Segment constraints automatically, and the historian's assets appear in the Relationship Graph the same day, not after a multi-month custom engineering engagement most organizations would never have funded in the first place.

WHAT THE SDK ACTUALLY SOLVES

- **Fragmented and inconsistent data classification.**
  Cloud providers, hypervisors, compute, network assets, and security platforms all describe the same category of asset differently. The configuration maps every provider object into a single, strongly typed platform catalog, imported directly from core WanAware platform libraries, so the central asset graph stays structurally clean no matter how many vendors feed it.

- **Conflicting and overlapping asset data.**
  The same physical server reported by AWS, CrowdStrike, and ServiceNow should be one record, not four, and when two tools disagree on a hostname or patch level, something has to decide which one wins. match-config.yaml defines deterministic correlation heuristics, MAC

addresses, serial numbers, BIOS UUIDs, cloud resource IDs, and survivorship-config.yaml sets attribute-level precedence, so network routing interfaces trust the network controller and host patch levels trust the EDR agent, by rule, not by guesswork.

- **Unsafe AI code generation.**
  Unconstrained generative AI introduces hallucinated dependencies, unvetted mappings, and broken type signatures. The SDK confines AI-assisted changes to designated provider directories, forces every AI-proposed mapping to confirmed: false until a human reviews it, and requires an explicit justification code, drawn from a fixed enumeration such as TRANSIENT_METRIC or DEPRECATED_UPSTREAM, for any field a generation run leaves unmapped.

- **Ingestion fragility and upstream API volatility.**
  Hand-written connectors tend to fail silently when a vendor quietly changes a response schema. Integrators built with the SDK carry standard contract tests and recorded-replay validation, so a schema shift is caught in development or CI, not discovered months later as stale data in the platform.

WHAT CHANGES, IN NUMBERS

Architectural capability	Bespoke integration engineering	Integrator SDK generation
Initial service scaffolding	80 to 120 engineering hours per connector.	Sub-60 minutes via declarative gen.
Transport and concurrency	Hand-rolled goroutines, manual channel locking.	Standardized worker pools, managed WebSocket handling.
Large-payload handling	Custom disk caching or memory-heavy serialization.	Integrated chunked cloud storage upload pipeline.
Schema registration	Manual schema authoring and CMDB migrations.	JSON configs embedded in binary, auto-registered at startup.
Code maintenance lifecycle	Manual refactor across every connector when a core API changes.	Re-run gen; SDK-owned files update, custom logic untouched.

None of this is a shortcut taken at the expense of rigor. It is the removal of repetitive engineering work that was never where the real value lived in the first place.

THE FOUR-STAGE QUALITY GATE PIPELINE

Every integrator, whether a human wrote it or an AI agent generated it, has to clear the same verify pipeline before it is considered shippable. The SDK calls this pipeline the platform's standardized definition of done, and it runs as four sequential, deterministic checkpoints. AI agent execution is held to strict iteration caps throughout, so a resolving error cannot spiral into an infinite loop.

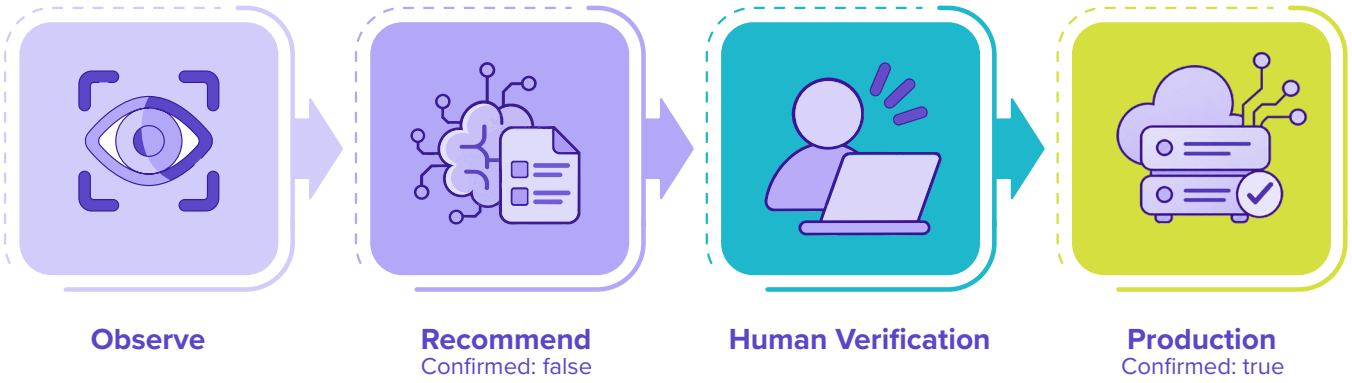
Stage	What runs	What it guarantees
1. Compile	Go toolchain (go build)	Zero syntax errors, valid types, complete dependency resolution.
2. Contract tests	Native test suite (go test)	Provider response parsers conform exactly to platform ingestion interfaces.
3. Response replay	Replay engine (internal/replay)	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.

GOVERNANCE BUILT FOR AI-ACCELERATED ENGINEERING

The SDK governs its own AI assistance the same way the rest of the WanAware platform governs automation: Observe, then Recommend, then a mandatory human checkpoint before anything reaches production. At the Observe level, the SDK inspects upstream structures and reports unmapped fields without changing anything. At the Recommend level, AI proposes property roles and subtypes, always written as confirmed: false. A human engineer must explicitly elevate a proposal to confirmed: true before it can pass the final shipping check, and the SDK will not let that check pass otherwise.

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.



Six tools, one MCP server

To work natively inside modern AI development environments, the SDK implements a Model Context Protocol server, reachable through integrator-sdk mcp, exposing six structured tools to any MCP-compliant assistant, including Claude Code, Cursor, and Windsurf: catalog, for programmatic access to the closed platform vocabulary; validate, for immediate schema checking; status, for visibility into unmapped fields and pending proposals; gen, to trigger deterministic compilation; patch, for controlled, auditable edits to specific configuration blocks; and shipcheck, to confirm a build is ready to ship. Custom Claude Code slash commands, /new-integrator and /fill-mapping-config, scaffold a clean provider project and auto-populate property roles from raw API responses, and a comprehensive AGENTS.md specification keeps non-Claude agents fully interoperable with the same workflow.

MEASURING WHETHER THE AI MAPPING CAN BE TRUSTED

A governance process is only as good as the thresholds it enforces. The SDK's internal evaluation harness, internal/eval, scores every AI-suggested mapping against an approved, human-validated reference answer set derived from the production-grade AWS integrator, across four statistical dimensions: subtype precision, how accurately assigned resource subtypes match the golden answer set; role assignment fidelity, whether property roles are classified correctly; catalog integrity, confirming zero hallucinated or out-of-catalog values; and run-to-run determinism, requiring near-identical outputs across repeated generation runs on the same upstream schema.

Metric	Required threshold
Resource subtype precision	≥ 90% exact match against the verified reference set
Property role precision	≥ 85% exact match against verified reference assignments
Catalog validation errors	Exactly 0 invalid or out-of-catalog values permitted
Run-to-run reproducibility	≥ 95% statistical agreement across repeated runs

In the interest of transparency: these thresholds govern internal engineering workflows today. Until they are formally validated and published through blind benchmark testing, AI-assisted mapping generation remains an internally gated capability rather than a publicly published accuracy claim, exactly the kind of discipline that keeps an automated mapping from quietly corrupting a downstream dependency graph or compliance attestation.

WHO THIS CHANGES THINGS FOR



Managed Service Providers

MSPs managing multi-tenant portfolios run into the same wall over and over: one client on Cisco Meraki and Fortinet, another on AWS and CrowdStrike, each requiring its own custom connector. The SDK lets MSPs build connectors for specialized or legacy client technologies in hours, cutting new-client infrastructure discovery from weeks to minutes and turning integration delivery from a project-based cost center into a recurring, continuous-visibility service.



Enterprises undergoing mergers and acquisitions

Acquiring IT organizations inherit heterogeneous, unmapped estates: overlapping private IP spaces, uninventoried cloud subscriptions, duplicate toolsets. The SDK lets corporate IT teams build bespoke connectors for an acquired entity's niche systems quickly, while the matching config and survivorship-config.yaml automate deduplication across the merged estate, surfacing redundant licensing and zombie cloud infrastructure within the first 90 days post-transaction.



Technology partners and ISVs

Software vendors need deep, bi-directional integration with asset intelligence platforms to prove value to shared customers, but building and maintaining those connectors consumes product engineering time. The SDK handles WebSocket management, cloud storage transport, worker pools, and catalog registration, so ISV engineers focus only on proprietary authentication and data models, cutting certified connector time-to-market from months to days while the ISV retains full ownership of its own codebase.

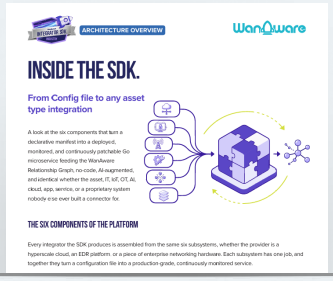


Highly regulated financial and healthcare institutions

Institutions governed by PCI DSS, SOC 2, and HIPAA need a continuous, defensible inventory, not a point-in-time spreadsheet. The SDK provides an auditable, deterministic framework for connecting clinical devices, EHR systems, mainframe databases, and

proprietary trading infrastructure into the Relationship Graph, with continuous, timestamped evidence mapped to NIST CSF 2.0 and HIPAA controls, replacing point-in-time audit preparation with standing, automated attestation.

Read the Integrator Architecture Overview here:



BUILT TRANSPARENTLY: CURRENT PLATFORM BOUNDARIES

Part of trusting an automated system is knowing exactly where its edges are today. The SDK's current release maturity is documented candidly rather than rounded up.

Capability	Current state	Where it is headed
Existing Integrators	Production ready; full crawls, auth, mapping, and contract tests functional	Reference implementation and evaluation baseline for every future provider
Real-time event dispatch	Architectural stub; interfaces generated, provider webhooks pending	Full streaming ingestion wired to provider event services such as AWS EventBridge
AI mapping accuracy benchmarks	Internal baseline gated	Public benchmark publication, then general availability for partners.
MCP server	Phase 1 active (catalog, validate, status); Phase 2 in review (gen, patch, shipcheck)	Phase 2 acceptance complete, enabling fully autonomous agent-driven scaffolding.

BUILT TRANSPARENTLY: CURRENT PLATFORM BOUNDARIES

~60-minute baseline

No-code, AI-augmented, from config file to compiling service

One Golden Record

Automated dedupe across every connected tool

Human-verified AI

Every proposal reviewed before it ships

Patch once, everywhere

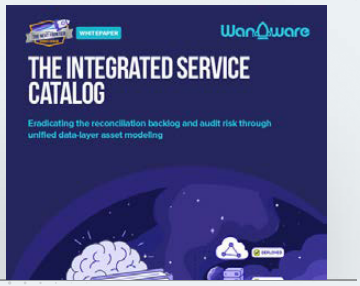
SDK-owned files update across the whole fleet

For enterprise customers, this means new cloud services, acquired systems, or niche vendor platforms can be folded into a central, deduplicated asset inventory within hours instead of quarters. For MSPs, GSIs, and ISVs, it means connector engineering stops being a speculative, multi-month cost center and becomes a predictable, fixed-fee deliverable, freeing engineering capacity for the domain-specific capability that actually differentiates their business.

CONCLUSION: INTEGRATION AS A DELIVERABLE, NOT A LIABILITY

Every hand-built connector is a small, compounding liability: more code to maintain, more drift to chase, more silent failure modes waiting for an upstream API to change without warning. The Integrator SDK does not remove engineering judgment from that work. It removes the parts that never required it, so the judgment that is left goes toward the authentication flow and data model that are genuinely specific to the provider in front of you, and toward the human review that keeps AI acceleration honest.

Turn your next integration backlog into a no-code, 60-minute build.



READY FOR LIFTOFF?

Start a free trial or see a demo at wanaware.com.

FREE TRIAL SEE A DEMO

WanAware delivers Actionable Observability and Asset Inventory Management for modern, connected enterprise environments. By combining real-time asset discovery with an intelligent, live dependency relationship graph, WanAware helps infrastructure, security, and reliability teams eliminate blind spots, drastically reduce alert noise, and prevent critical enterprise downtime.