PRODUCT TECHNICAL SERVICE DESCRIPTION

WHITEPAPER

WanAware

# WANAWARE INTEGRATOR SDK

## Automated Platform Integration SDK for IT, AI, OT, IoT and Cloud Infrastructure



# TABLE OF CONTENTS


**EXECUTIVE SUMMARY AND THE INTEGRATION GAP** 3
- The Core Integration Crisis and the Ecosystem Imperative 3
- WanAware Integrator SDK: Turning Integrations into Predictable Deliverables 4

**THE PARADIGM SHIFT: FROM MANUAL CODING TO AI-ASSISTED INTEGRATOR GENERATION** 4
- The Architectural Failure of Bespoke Integrations 4
- The Dynamic Code Generation Architecture 5
- Deterministic Scaffolding and Safe Regeneration Mechanics 5

**WHAT THE INTEGRATOR SDK DELIVERS IN UNDER 5 MINUTES** 6
- Rapid Baseline Service Construction 6
- Operational Comparison: Bespoke Engineering versus SDK Generation 6

**THE CHALLENGES THE INTEGRATOR SDK SOLVES** 7
- Fragmented and Inconsistent Data Classification 7
- Conflicting and Overlapping Asset Data 8
- Unsafe AI Code Generation and Maintenance Drift 8
- Ingestion Fragility and Upstream API Volatility 9

**ARCHITECTURE AND METHODOLOGY: HOW THE SDK WORKS8** 10
- Continuous AI-Assisted Architecture and Context Injection 10
- The Four-Stage Quality Gate Pipeline 10

**CORE COMPONENTS OF THE PLATFORM** 11
- 1. Code Generation Engine (gen) 11
- 2. Asset Categorization and Tier-0 Configuration Engine 11
- 3. Local Validation and Quality Assurance Gates 12
- 4. AI-Agent Orchestration and Model Context Protocol (MCP) Server 12
- 5. Automated Accuracy Measurement and Reference Answer Evaluation 13
- 6. Operational Lifecycle and Platform Telemetry Integration 13

**INTELLIGENCE, AUTOMATION, AND ACTION LEVELS** 14
- Rigorous Accuracy Thresholds and Benchmark Targets 14
- Governance Framework: Observe, Recommend, and Human-in-the-Loop Safeguards 14

**INDUSTRY USE CASES AND MEASURABLE OUTCOMES** 15
- Managed Service Providers (MSPs) 15
- Enterprises Undergoing Mergers and Acquisitions (M&A) 15
- Technology Partners and Independent Software Vendors (ISVs) 16
- Highly Regulated Financial and Healthcare Institutions 16

**SUMMARY OF OUTCOME-BASED BENEFITS** 17
- Quantified Customer Value Matrix 17
- Strategic Partner Commercial and Operational Value Matrix 17

**DEPLOYMENT, SERVICE LEVELS, AND CURRENT LIMITS** 18
- Deployment Framework and Operational Baseline 18
- Current Platform Boundaries and Architectural Limits 19

**WHY THE INTEGRATOR SDK?** 19

# EXECUTIVE SUMMARY AND THE INTEGRATION GAP

## The Core Integration Crisis and the Ecosystem Imperative

Modern enterprise infrastructure has expanded beyond the perimeter of traditional data centers into distributed, multi-cloud platforms, ephemeral container environments, software-defined networks, and specialized security architectures. While this evolution accelerates functional velocity, it introduces a severe operational bottleneck: telemetry fragmentation. Enterprise technology ecosystems routinely operate dozens of disconnected monitoring, security, cloud, and networking tools, each maintaining a partial, isolated perspective of the physical and virtual estate.

Historically, unifying these disparate telemetry streams into a centralized asset intelligence platform required building bespoke integration connectors for each discrete endpoint API. This integration methodology introduces systemic friction. Software engineering teams repeatedly construct boilerplate transport mechanisms, custom pagination handlers, error-retry loops, authentication routines, and transient data formatting layers. The resulting integration estate is fragile, maintenance-heavy, and prone to silent failures whenever upstream third-party APIs modify their response schemas or authentication parameters. For enterprise customers, this crisis manifests as delayed visibility, stale configuration databases, and prolonged blind spots across security and infrastructure estates.

### THE DATA BLACK HOLE

Surveys indicate that fewer than 40% of organizations trust their current asset data, while more than half report that inaccurate inventories directly triggered an outage, security vulnerability, or performance failure over the preceding year.



Up to 25% of enterprise infrastructure expenditure is wasted on untracked compute instances, unattached storage volumes, and abandoned IP allocations. For technology partners—including Managed Service Providers (MSPs), Global Systems Integrators (GSIs), and Independent Software Vendors (ISVs)—the manual engineering overhead of connector development introduces severe commercial liabilities. Engineering capacity is consumed by repetitive low-level plumbing rather than domain-specific capability development, leading to elongated time-to-market and inflated delivery costs.

## WanAware Integrator SDK: Turning Integrations into Predictable Deliverables

The **WanAware Integrator SDK** transforms integration development from a manual software development effort into a deterministic, configuration-driven, and AI-accelerated process. Implemented as a Go command-line interface (CLI) coupled with structured rules and playbooks for artificial intelligence coding agents, the SDK standardizes how integrators are scaffolded, implemented, validated, and shipped. Within the WanAware architectural model, an integrator operates as a dedicated microservice engineered to extract asset inventory from external providers—such as hyperscale clouds, endpoint detection and response (EDR) platforms, and enterprise networking hardware—and stream normalized asset payloads into the WanAware Asset Inventory Management (AIM) platform.

For customers, this standardization guarantees uniform data fidelity, automated deduplication, and reliable ingest cadences across all monitored platforms. For partners, the SDK converts custom connector engineering from a speculative, multi-month expenditure into a predictable, repeatable process completed in hours or days, facilitating rapid monetization of integration offerings and opening recurring managed-service revenue streams.

The architectural premise of the SDK is complete separation of operational boilerplate from provider-specific domain logic. By automatically generating transport plumbing, concurrency management, WebSocket interfaces, object storage transfers, and schema registrations, the SDK restricts the developer or AI agent's responsibility to writing targeted authentication and data extraction routines.

# THE PARADIGM SHIFT: FROM MANUAL CODING TO AI-ASSISTED INTEGRATOR GENERATION

## The Architectural Failure of Bespoke Integrations

Bespoke integration engineering suffers from architectural degradation over time. When developers hand-craft individual connectors, architectural drift inevitably occurs across the integration catalog. Hand-rolled implementations of exponential backoff, rate-limit throttling, and status handling introduce unpredictable failure modes across services.

Disparate developers apply inconsistent logic when deriving unique identifiers from upstream payloads, corrupting downstream asset correlation engines and leaving configuration databases riddled with duplicate entries. Furthermore, hand-written worker pools frequently mishandle memory allocation or thread synchronization under high API volume, triggering out-of-memory errors during large inventory collections. This technical divergence creates an unsustainable maintenance burden where every minor change to an upstream vendor API triggers extensive, brittle refactoring across the integration fleet.

## The Dynamic Code Generation Architecture

The WanAware Integrator SDK resolves these issues through declarative compilation via its code generation (gen) subsystem. Integrator developers describe provider endpoints, target resources, and extraction behaviors within a single configuration manifest: sdk-config.yaml. Using this declared specification, the SDK then deterministically generates source code boilerplate and scaffolding which compiles into a deployable Go microservice.

The generated codebase automatically implements five universal integration patterns, removing the need for manual implementation:

1. First, it implements standardized cursor and offset pagination loops that systematically traverse provider API responses without buffer overflows or infinite loops.
2. Second, it incorporates deterministic intercepts for missing or deleted upstream objects, preventing integration crashes and triggering appropriate tombstone lifecycles within the asset graph.
3. Third, it applies uniform string parsing algorithms that sanitize provider-native identifiers into globally unique platform external IDs.
4. Fourth, it injects dynamic runtime filters that enforce selective resource discovery, preventing unwanted API consumption or out-of-scope asset gathering.
5. Finally, it establishes decoupled notification structures that transmit real-time change events directly to WanAware event streams.

## Deterministic Scaffolding and Safe Regeneration Mechanics

A critical design requirement of enterprise-grade code generation is safe iterative execution.



### FAST FACTOID

Traditional code scaffolding tools generate a static project template once; any subsequent re-run overwrites manual customizations or requires error-prone code diffing. The WanAware Integrator SDK enforces a strict, two-tier file ownership boundary.

All infrastructure files—including scaffolding entry points, configuration readers, WebSocket connection managers, storage upload modules, telemetry bindings, application container files, and Makefiles—are designated as SDK-owned. These files are safely overwritten on every execution of the gen command, guaranteeing that upstream SDK enhancements, security patches, and performance optimizations can be applied across the entire connector catalog instantly. Conversely, provider-specific authentication handlers, custom API query strings, and payload parsing methods are isolated into dedicated stub files created once by the generator and left untouched during subsequent generation cycles.

This separation yields substantial commercial value. Customers benefit from an integration fleet that can be patched against low-level operational vulnerabilities without breaking domain-specific data parsing.

Partners can continuously regenerate and update their commercial connectors against newer WanAware SDK releases without risking regressions in their custom client code, protecting their intellectual property and engineering investment over multi-year software lifecycles.

## WHAT THE INTEGRATOR SDK DELIVERS IN UNDER 5 MINUTES

### Rapid Baseline Service Construction

Traditional enterprise integration development requires days of initial scaffolding before the first external API call is executed. Developers must configure dependency managers, container specifications, network abstractions, event consumers, and storage pipeline endpoints.




The WanAware Integrator SDK collapses this setup lifecycle to **under five minutes.** By executing integrator-sdk gen against a defined sdk-config.yaml, the developer or an AI coding agent transitions from an empty directory to a fully compiling, deployable Go service. During validation runs against the AWS provider catalog, autonomous AI agents utilizing the SDK successfully progressed from raw service specifications to fully compiling Go services passing automated contract tests in four to five minutes per service, requiring zero human intervention.

### Operational Comparison: Bespoke Engineering versus SDK Generation

The operational and financial differences between legacy bespoke integration development and the WanAware Integrator SDK architecture are detailed below:

| Architectural Capability | Bespoke Integration Engineering | WanAware Integrator SDK Generation | Operational and Strategic Impact |
|---|---|---|---|
| **Initial Service Scaffolding** | 80–120 engineering hours per connector. | Sub-60 minutes via declarative gen execution. | Accelerates time-to-market by orders of magnitude for partners and internal teams. |
| **Transport & Concurrency** | Bespoke goroutines, manual channel locking, custom HTTP clients. | Standardized worker pools, managed WebSocket handlers, native rate limiting. | Eliminates concurrency deadlocks and memory leaks in production runtime. |
| **Large-Payload Handling** | Custom disk caching or memory-intensive network serialization. | Integrated Google Cloud Storage (GCS) chunked file upload pipeline. | Prevents memory exhaustion during multi-gigabyte infrastructure crawls. |

| Architectural Capability | Bespoke Integration Engineering | WanAware Integrator SDK Generation | Operational and Strategic Impact |
|---|---|---|---|
| **Platform Telemetry** | Inconsistent logging frameworks, sporadic or absent metrics. | Out-of-the-box OpenTelemetry tracing and structured JSON logging. | Guarantees continuous operational observability across complex customer environments. |
| **Schema Registration** | Manual schema authoring, API registration scripts, CMDB migrations. | JSON configs embedded in binary and auto-registered at startup. | Eliminates manual database modifications and configuration drift. |
| **Code Maintenance Lifecycle** | Manual refactoring across every connector codebase when core platform APIs evolve. | Re-run integrator-sdk gen; SDK-owned files update while custom logic is untouched. | Dramatically lowers total cost of ownership (TCO) across multi-year software lifecycles. |

For customers, this rapid construction timeline ensures that newly introduced cloud services, edge devices, or proprietary infrastructure can be incorporated into their central asset inventory within hours of operational deployment. For partners, it compresses billable development cycles, enabling high-margin fixed-fee integration delivery models rather than protracted time-and-materials engagements.

# THE CHALLENGES THE INTEGRATOR SDK SOLVES

## Fragmented and Inconsistent Data Classification

When infrastructure asset data is pulled from disparate vendors, data structures diverge significantly. AWS describes compute nodes through nested EC2 instance specifications; VMware categorizes virtual machines through distinct managed object references; network switches identify ports via interface indexes; and security platforms identify assets via proprietary agent GUIDs. Without strict classification standards, downstream consumers must write complex transformation filters for every query.

The SDK resolves classification divergence through configuration specifications. In mapping-config.yaml, integrators map raw provider objects into a closed, strongly typed platform catalog imported directly from core WanAware platform libraries. Resources are strictly typed into base entities (such as Asset, IP Block, or Network Interface) and qualified by platform-governed subtypes.

Customers benefit because the central asset graph remains structurally clean. Security, FinOps, and network engineering teams can query a uniform schema without maintaining custom translation logic for each vendor. Partners benefit from the complete elimination of data modeling guesswork. Integrator developers reference a static, validated catalog, preventing integration submission rejections caused by incompatible categorization labels.

## Conflicting and Overlapping Asset Data

A common operational issue in enterprise management is asset duplication. When an AWS EC2 instance runs a CrowdStrike Falcon sensor, runs a virtual Cisco Meraki security appliance, and generates records inside ServiceNow, legacy systems frequently create four distinct asset records for that single physical or virtual asset. Conversely, if two tools report conflicting values for hostnames or operating system patch states, standard repositories have no programmatic method to determine ground truth.

The SDK resolves these identity and conflict challenges through two native configuration files. The first, match-config.yaml, defines deterministic correlation heuristics (such as matching MAC addresses, serial numbers, BIOS UUIDs, or cloud provider resource IDs) that instruct the central WanAware engine on how to recognize the same asset across disparate collection streams. The second, survivorship-config.yaml, dictates strict attribute-level precedence rules. For example, network routing interfaces prioritize telemetry from Cisco network controllers, while host patch revisions prioritize data from EDR agents.

This architecture provides immediate value to customers by eliminating duplicate asset counts and phantom infrastructure records, yielding an authoritative Golden Record. Partners can integrate their specialized solutions into environments running incumbent tools without risk of corrupting existing customer configurations.

## Unsafe AI Code Generation and Maintenance Drift

While generative AI models accelerate software development, their unconstrained deployment introduces significant risks to enterprise codebases: hallucinated dependencies, insecure file system operations, unvetted mappings, and broken type signatures.

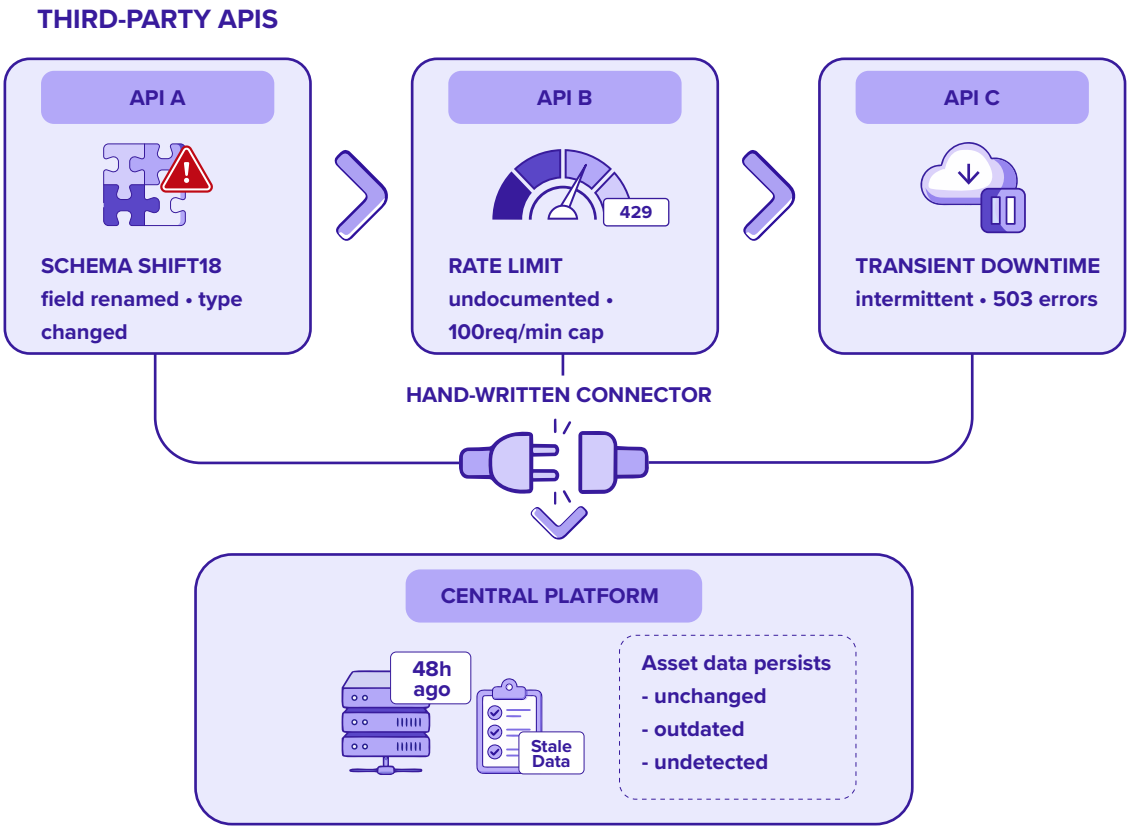
The WanAware Integrator SDK introduces strict, programmatic guardrails enforced directly within the CLI and Model Context Protocol (MCP) tooling. AI-assisted modifications are cryptographically and programmatically confined strictly to designated provider directories, preventing unauthorized modifications to platform core modules or local system files.

Furthermore, AI agents are programmatically restricted to setting mapping confirmation flags to confirmed: false. A human engineer must review and explicitly elevate proposals to confirm: true, preventing unvetted automated interpretations from reaching production environments. Finally, to eliminate incomplete integrations, any upstream API field that is left unmapped must be accompanied by an explicit justification code selected from a fixed, platform-enforced enumeration (such as TRANSIENT_METRIC, DEPRECATED_UPSTREAM, or REDUNDANT_IDENTIFIER).

For customers, these guardrails ensure that every integration deployed into their network meets enterprise reliability and safety standards. For partners, it allows engineering leadership to safely deploy autonomous AI coding agents to accelerate development without risking compliance violations or codebase corruption.

## Ingestion Fragility and Upstream API Volatility

Third-party APIs frequently introduce subtle schema shifts, undocumented rate limits, and transient availability drops. Hand-written connectors often fail silently when these events occur, causing stale asset data to persist undetected in the central platform.




The Integrator SDK addresses this through standard contract tests, recorded replay validations, and native OpenTelemetry tracing. Integrations compiled with the SDK log standardized health metrics, crawl duration stats, and payload volume telemetry. If an upstream vendor alters an API response, the integrator's local contract test suite captures the divergence immediately during development or CI/CD pipelines, preventing faulty builds from shipping.

# ARCHITECTURE AND METHODOLOGY: HOW THE SDK WORKS

## Continuous AI-Assisted Architecture and Context Injection

The architecture of the WanAware Integrator SDK is designed to support both human developers and autonomous AI coding agents (such as Claude Code, Cursor, Windsurf, and OpenAI Codex). It achieves this through structured context injection contained within the SDK's codegen/ ecosystem.

### NEW HORIZONS



These machine-readable rules are embedded directly into the binary and pinned to the SDK release version, ensuring agents utilize the exact APIs supported by the running platform version without requiring a full local checkout of the main repository.

Rather than relying on generic Large Language Model (LLM) training weights, the SDK provides agents with targeted contextual assets. It incorporates curated provider research notes detailing specific API idiosyncrasies, rate limit envelopes, and authentication flows for AWS, Azure, Google Cloud Platform (GCP), CrowdStrike, Fortinet, and Cisco Meraki. Complementing these notes are codegen playbooks that dictate how an agent must construct discovery loops, format request headers, and translate complex nested structures into WanAware platform definitions.

## The Four-Stage Quality Gate Pipeline

To enforce production readiness across both human-authored and AI-generated codebases, the SDK implements verify: a rigid, four-stage validation gate that serves as the platform's standardized definition of done. When autonomous AI agents execute integration workflows, their execution is monitored under strict iteration caps, preventing infinite loop generation when resolving code errors.

The pipeline executes sequentially through four deterministic checkpoints. It begins with compilation, transitions through contract tests to validate structural interfaces, proceeds into a replay engine that evaluates recorded provider responses, and concludes with a comprehensive shipping check. The technical mechanics and quality criteria across each stage are outlined below:

| Pipeline Stage | Subsystem Executed | Technical Validation Criteria | Architectural and Quality Objective |
|---|---|---|---|
| **Stage 1:** Compile | Go Toolchain (go build) | Absolute type checking, package validation, and compilation of the binary. | Ensures zero syntax errors, valid type assignments, and complete dependency resolution. |
| **Stage 2:** Contract Tests | Native Testing Suite (go test) | Verifies that all provider response parsers conform to platform ingestion interfaces. | Guarantees that data structural models match platform-expected schemas precisely. |
| **Stage 3:** Response Replay | Replay Engine (internal/replay) | Replays recorded real-world provider API payloads against the compiled parsing stubs. | Confirms resilience against real-world payload variations and structural edge cases. |
| **Stage 4:** Shipcheck | integrator-sdk shipcheck | Comprehensive check: go vet, code formatting (gofmt), unit tests, Tier-0 catalog validation, and idempotent regeneration checks. | Ensures pristine code style, zero catalog violations, and absolute stability under subsequent gen runs. |

This quality pipeline provides substantial value to customers by ensuring that all third-party and custom-built connectors maintain uniform enterprise-grade reliability, eliminating platform outages caused by poorly tested integrations. For partners, the deterministic gate automates the quality assurance lifecycle, allowing developers to validate production readiness locally before submitting code to customer environments.

## CORE COMPONENTS OF THE PLATFORM

### 1. Code Generation Engine (gen)

The gen subsystem serves as the core build engine of the SDK. Driven by sdk-config.yaml, it compiles the primary operational scaffold of the Go microservice. This includes managed WebSocket client interfaces capable of receiving platform commands (such as ad-hoc crawl requests and configuration refreshes) and streaming diagnostic health metrics. It packages integrated multi-part upload modules that serialize, compress, and stage large asset payloads, automatically triggering platform ingest webhooks upon upload completion.

The engine also builds configurable worker pools that execute parallel resource crawls while strictly respecting upstream provider API rate limits. Embedded OpenTelemetry tracing collectors and structured logging packages are woven into the codebase, emitting unified telemetry across all execution paths.

### 2. Asset Categorization and Tier-0 Configuration Engine

The categorization engine translates proprietary vendor metadata into the standardized WanAware Relationship Graph. It is driven by three Tier-0 configuration manifests. The primary entity structure is defined in mapping-config.yaml, which establishes the base platform entity, its qualified subtype, and the operational role of

each individual field, such as Primary Identifier, IP Address, Hostname, MAC Address, or Lifecycle Tag. Identity correlation logic is codified in match-config.yaml, which defines multi-attribute priority queues for asset reconciliation across disparate sources.

Finally, attribute conflict resolution hierarchies are formalized in survivorship-config.yaml, ensuring authoritative sources supersede secondary collectors.

These configurations are compiled directly into static JSON structures embedded into the compiled Go binary. Upon service initialization, the integrator transmits these schemas to the WanAware platform via automated registration endpoints, eliminating manual database schema updates and ensuring zero-downtime operational enablement.

## 3. Local Validation and Quality Assurance Gates

To accelerate the developer feedback loop, the SDK provides offline validation commands that run locally without active platform connectivity. The integrator-sdk validate command parses local configuration manifests against the central platform's closed catalog schema, flagging illegal subtypes, invalid role assignments, and broken references. The integrator-sdk status utility emits a detailed operational dashboard listing every upstream resource that lacks mapping coverage, alongside an itemized inventory of AI-suggested mappings that await human confirmation.



### FAST FACTOID

The integrator-sdk shipcheck command acts as the final pre-deployment gate, ensuring that the project adheres to all compilation, vetting, testing, schema validation, and regeneration safety standards.

## 4. AI-Agent Orchestration and Model Context Protocol (MCP) Server

To provide deep integration with modern development environments, the SDK implements an integrated Model Context Protocol (MCP) server accessible via integrator-sdk mcp. This server exposes SDK capabilities natively as structured tools to any MCP-compliant AI assistant, including Claude Code, Cursor, and Windsurf.

The MCP server exposes six primary tools to the autonomous agent: catalog provides programmatic access to the closed platform vocabulary and valid subtype hierarchies; validate executes immediate schema checking against working manifests; status informs the agent of remaining unmapped fields and pending proposal gates; gen enables the agent to trigger deterministic project compilation; patch permits controlled, auditable modifications to specific configuration blocks; and shipcheck validates whether the agent's working modifications meet shipping criteria.

The SDK packages custom Claude Code slash commands, specifically /new-integrator to scaffold a clean provider project and /fill-mapping-config to autonomously populate property roles from raw API responses. For non-Claude environments, a comprehensive AGENTS.md context specification ensures full interoperability across diverse agent platforms.

## 5. Automated Accuracy Measurement and Reference Answer Evaluation

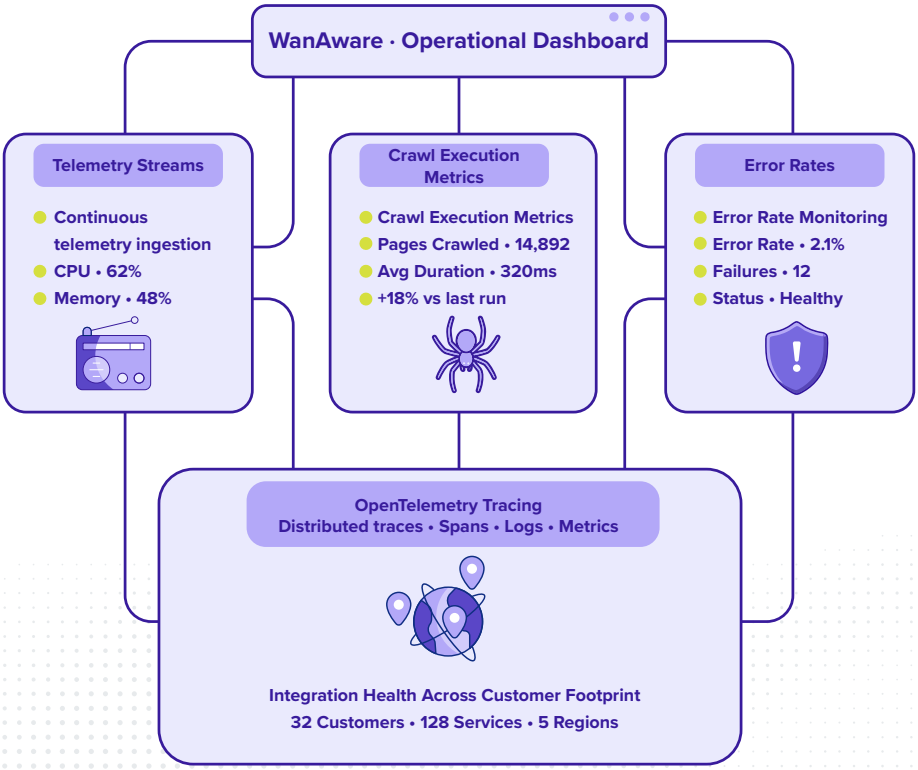
To prevent the degradation of asset classification quality, the SDK incorporates an internal evaluation subsystem located in internal/eval. This harness evaluates AI-suggested mappings against an approved, human-validated reference answer set derived from the production-grade AWS integrator.

The subsystem scores proposed mappings across four statistical dimensions: subtype precision measures the statistical accuracy of assigned resource subtypes against the golden answer set; role assignment fidelity evaluates whether property roles are correctly classified; catalog integrity verifies that zero hallucinated or out-of-catalog values exist within the generated schema; and run-to-run determinism assesses semantic stability by requiring near-identical mapping outputs across repeated generation executions on identical upstream schemas.

## 6. Operational Lifecycle and Platform Telemetry Integration

Integrators generated by the SDK are fully integrated into the WanAware operational fleet. Each compiled microservice natively supports containerized deployment patterns via the generated Dockerfile and Makefile, connecting seamlessly into Google Cloud Kubernetes Engine environments.

Telemetry streams, crawl execution metrics, and error rates are continuously emitted to WanAware's operational dashboards via OpenTelemetry tracing, providing clear visibility into integration health across the entire customer footprint.



# INTELLIGENCE, AUTOMATION, AND ACTION LEVELS

## Rigorous Accuracy Thresholds and Benchmark Targets

To maintain enterprise-grade asset data fidelity, the WanAware Integrator SDK establishes strict mathematical thresholds governing automated intelligence. Machine learning algorithms and LLM agents utilized in mapping generation must satisfy four mandatory baseline targets:

- Resource Subtype Precision: Greater than or equal to 90% exact match against the verified reference answer set.
- Property Role Precision: Greater than or equal to 85% exact match against verified reference property assignments.
- Catalog Validation Errors: Exactly 0 invalid or out-of-catalog values permitted.
- Run-to-Run Reproducibility: Greater than or equal to 95% statistical agreement across repeated execution cycles on the same source schema.

By enforcing these statistical baselines, WanAware eliminates the risk of hallucinated attributes corrupting downstream dependency graphs, financial calculations, and compliance attestations. Until these metrics are formally validated and published in blind benchmark testing, AI-assisted mapping generation remains restricted to internal engineering workflows.

## Governance Framework: Observe, Recommend, and Human-in-the-Loop Safeguards

In alignment with the core WanAware platform governance model, the Integrator SDK operates under an Action Level framework designed to guarantee operational safety and human oversight. The governance progression moves sequentially from non-invasive data observation to AI-driven recommendation, passing through a mandatory human verification checkpoint before artifacts can be compiled into production binaries.

At the Observe Level, which represents the default state, the platform and SDK extract raw telemetry, inspect upstream JSON structures, and present unmapped field status via integrator-sdk status without executing configuration changes or altering platform state.

At the Recommend Level, the AI codegen subsystem parses observed schemas and generates proposed property roles and subtypes within mapping-config.yaml. Crucially, the tooling enforces a hardcoded safeguard: all AI proposals are written with the parameter confirmed: false.

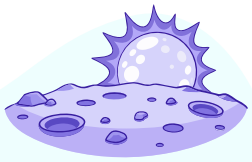
The Human-in-the-Loop Verification phase acts as an immutable boundary: the SDK prevents any configuration file containing confirmed: false from passing the final shipcheck gate. A human engineer must explicitly review each proposed mapping and elevate the status to confirmed: true. Furthermore, any unmapped fields must contain a validated justification code selected from the closed catalog enumeration. Only after satisfying these criteria can the project proceed to production runtime compilation and automated registration.

This governance model provides significant peace of mind. Customers are protected against unverified automated changes corrupting their asset intelligence. Partners maintain an auditable, compliant engineering trail proving that human domain experts verified all data transformations prior to deployment.

## INDUSTRY USE CASES AND MEASURABLE OUTCOMES

### Managed Service Providers (MSPs)

MSPs managing complex multi-tenant client portfolios struggle with disparate technology stacks across clients. One client may utilize Cisco Meraki for networking and Fortinet for perimeter defense, while another utilizes AWS cloud-native constructs and CrowdStrike EDR. Building and maintaining custom connectors for this diverse toolset historically eroded service margins.



#### NEW HORIZONS

By standardizing on the SDK, MSPs evolve from project-based integration delivery into recurring managed service providers, delivering continuous asset visibility, license optimization, and attack surface management.

The SDK allows MSPs to build custom connectors for specialized or legacy client technologies in hours. The resulting integrators run efficiently within client environments, streaming clean asset telemetry into the multi-tenant WanAware platform. MSPs eliminate manual client onboarding sweeps, reducing new-client infrastructure discovery from weeks to minutes.

### Enterprises Undergoing Mergers and Acquisitions (M&A)

Following corporate acquisitions, acquiring enterprise IT organizations inherit heterogeneous, unmapped infrastructure estates featuring overlapping private IP spaces, uninventoried cloud subscriptions, and duplicate toolsets.

Corporate IT teams leverage the Integrator SDK to rapidly develop bespoke connectors for the acquired entity's niche systems, proprietary databases, and private cloud platforms. The SDK's match-config.yaml and

survivorship-config.yaml enable automated deduplication across the merged entities. Enterprises identify redundant vendor licensing, discover zombie cloud infrastructure, and consolidate overlapping network domains, systematically recovering wasted capital within the first 90 days post-transaction.

## Technology Partners and Independent Software Vendors (ISVs)

Software vendors in the security, networking, and cloud management sectors require deep bi-directional integration with enterprise asset intelligence platforms to demonstrate product value to mutual customers. However, developing and maintaining proprietary connectors for external platforms consumes valuable product engineering resources.

The WanAware Integrator SDK allows ISVs to build, package, and commercially maintain their own official WanAware integrators. Because the SDK handles WebSocket management, GCS transport, worker pools, and catalog registration, ISV engineers focus solely on their proprietary authentication and data models.

> Time-to-market for certified ISV connectors is reduced from months to days. ISV partners retain complete ownership over their integration codebase while ensuring compatibility with WanAware's central platform releases.

## Highly Regulated Financial and Healthcare Institutions

Financial institutions (governed by PCI DSS and SOC 2) and healthcare organizations (governed by HIPAA) must maintain continuous, defensible inventories of all connected systems and data repositories. Stale spreadsheets and point-in-time audits expose these organizations to severe regulatory penalties and breach vulnerabilities.

The SDK provides an auditable, deterministic framework for connecting clinical devices, electronic health record (EHR) systems, mainframe databases, and proprietary trading infrastructure into the **WanAware Relationship Graph.** Integrations compiled via the SDK provide continuous, timestamped evidence chains demonstrating that all connected hardware and software assets are accounted for, categorized against NIST CSF 2.0 and HIPAA controls, and monitored for lifecycle changes. Point-in-time audit preparation is replaced with automated, continuous attestation.

# SUMMARY OF OUTCOME-BASED BENEFITS

## Quantified Customer Value Matrix

The direct strategic, operational, and financial outcomes realized by enterprise customers deploying integrators built with the SDK are synthesized below:

| Enterprise Value Dimension | Pre-SDK Operational State | Post-SDK Operational State | Quantified Customer Business Outcome |
|---|---|---|---|
| **Asset Visibility & Duplication** | Fragmented asset records across isolated consoles; up to 25% duplicate or ghost records. | Single authoritative Golden Record synthesized via automated match and survivorship rules. | Up to 25% infrastructure spend recovered; elimination of duplicate asset records across tools. |
| **Connector Deployment Speed** | 3 to 6 months required to scope, fund, and engineer custom enterprise connectors. | New custom connectors built, validated, and deployed in days via SDK automation. | Over 90% reduction in integration delivery timelines; immediate time-to-value for new tools. |
| **Audit & Regulatory Compliance** | Weeks-long manual spreadsheet reconciliation; point-in-time compliance snapshots. | Continuous, automated evidence attestation mapped directly to NIST CSF 2.0, SOC 2, and HIPAA. | Elimination of external audit findings; 100% auditable provenance on all inventory records. |
| **Operational Reliability** | Bespoke connectors crash due to unhandled rate limits, API shifts, and memory leaks. | Standardized Go worker pools, four-stage verification gates, and OpenTelemetry monitoring. | Prevention of ingestion outages; near-zero platform downtime caused by connector failures. |

## Strategic Partner Commercial and Operational Value Matrix

The direct commercial, engineering, and competitive advantages realized by technology partners, GSIs, and MSPs adopting the SDK include:

| Partner Value Dimension | Traditional Integration | WanAware Integrator SDK | Commercial and Operational |
|---|---|---|---|
| **Engineering Time-to-Market** | 80–120 billable hours per provider integration; heavy manual boilerplate development. | 60 minutes per service baseline generation; rapid completion of provider logic. | 80%+ reduction in non-billable development overhead; rapid fixed-price margin realization. |
| **Intellectual Property Ownership** | Fragmented custom scripts built for specific clients; difficult to reuse or commercialize. | Standardized, versioned Go binaries that partners can package, own, and maintain. | Creation of reusable IP; evolution into high-margin recurring managed integration services. |

| Partner Value Dimension | Traditional Integration | WanAware Integrator SDK | Commercial and Operational |
|---|---|---|---|
| **Quality Assurance Overhead** | Protracted manual testing and troubleshooting in live customer environments. | Local validation via validate, status, and four-stage automated verify gates. | Elimination of integration deployment regressions; instant local verification of schema compliance. |
| **AI Tooling Interoperability** | Ad-hoc, unguided prompting of commercial LLMs leading to hallucinated mappings and bugs. | Native MCP server integration, playbooks, and slash commands compatible with any agent.Native MCP server integration, playbooks, and slash commands compatible with any agent. | Safe, structured AI acceleration across development teams without vendor lock-in. |

## DEPLOYMENT, SERVICE LEVELS, AND CURRENT LIMITS

### Deployment Framework and Operational Baseline

Integrators generated by the WanAware Integrator SDK deploy as lightweight, containerized Go microservices designed to run on customer-managed infrastructure, edge gateways, or within the multi-tenant WanAware Platform Worker Fleet.

Operational performance conforms to strict baseline availability and freshness standards.

Telemetry reflects near real-time propagation of configuration updates, with discovery cycles executing via scheduled polling sweeps, ad-hoc administrator requests, or real-time event webhooks.

To ensure minimal impact on target infrastructure, integrators utilize credential-delegated, read-only API access (such as AWS IAM AssumeRole configurations), completely avoiding the installation of intrusive software agents on managed target nodes.

### MISSION CONTROL: BY THE NUMBERS

- MTTR reduced by **90%**
- Alert noise suppressed by up to **99.9%**
- **Zero** professional services cost



## Current Platform Boundaries and Architectural Limits

To maintain complete architectural transparency, the current operational boundaries and development roadmap of the WanAware Integrator SDK codebase are detailed in the capability maturity matrix below:

| Architectural Capability | Current Release Maturity State | Operational Impact and Production Boundaries | Forward Architectural Trajectory |
|---|---|---|---|
| **AWS Provider Integration** | Production Ready. | Full end-to-end operational crawls, authentication, mapping, and contract tests fully functional. | Serves as the primary reference implementation and evaluation baseline for all future providers. |
| **Secondary Cloud & Security Providers** | Scaffolding / Playbook Phase. | Azure, GCP, CrowdStrike, Fortinet, and Meraki research notes and playbooks exist; credential wiring is underway. | Platform-side credential onboarding templates transitioning from placeholder content to production. |
| **Real-Time Event Dispatch** | Architectural Stub. | Event handling interfaces are generated in code; provider-specific webhook listeners require completion. | Full real-time streaming ingestion being wired to provider-specific event services (e.g., AWS EventBridge). |
| **AI Mapping Accuracy Benchmarks** | Internal Baseline Gated. | Evaluation harness (internal/eval) and targets approved; external availability gated pending blind runs. | Public publication of benchmark accuracy scores; subsequent general availability release for partners. |
| **Model Context Protocol (MCP) Server** | Phase 1 Active; Phase 2 In Progress. | Phase 1 tools (catalog, validate, status) functional; Phase 2 tools (gen, patch, shipcheck) in code review. | Completion of Phase 2 acceptance runs to enable fully autonomous agent-driven scaffolding workflows. |

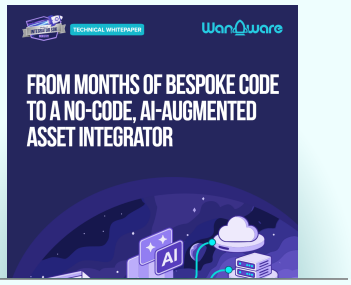
## WHY THE INTEGRATOR SDK?

Modern enterprise IT and cybersecurity operations require an accurate, continuously refreshed understanding of every connected infrastructure asset. When integrations are built manually, this foundational visibility degrades under the weight of engineering backlogs, brittle custom code, divergent data schemas, and mounting technical debt.

The WanAware Integrator SDK resolves this challenge by turning integration development into a deterministic, automated, and AI-accelerated discipline. By automating common operational patterns—pagination, error recovery, large-payload uploads, WebSocket communication, and schema registration—the SDK allows human engineers and AI coding agents to focus exclusively on domain logic.

**See how WanAware does it.**

**Read the white paper here:**



Coupled with strict Tier-0 catalog enforcement, automated match and survivorship configurations, and rigorous four-stage quality gates, the SDK guarantees that every connector delivers clean, deduplicated, and auditable data into the WanAware Relationship Graph.

For enterprise customers, this delivers comprehensive visibility, eliminates financial waste, and establishes continuous regulatory compliance. For partners, it unlocks rapid time-to-market, protects commercial margins, and provides the ability to independently scale and monetize custom integration ecosystems with complete operational confidence.

## 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.