

Agentic BSS

Abstract

Telecom BSS is under pressure from 5G/IoT service complexity, enterprise-grade expectations, and rising operational cost. While many platforms have added AI assistants, most implementations remain add-ons that do not fundamentally change how offers, orders, care, and billing assurance are executed and governed.

Cloudnet.ai defines **Agentic BSS** as an AI-native BSS operating model where specialized agents perform task execution through controlled APIs and event triggers, under telco-grade safeguards: **bounded autonomy**, human-in-the-loop approvals where required, policy enforcement, traceable execution, and rollback. This enables BSS to evolve from passive record-keeping and manual orchestration to **guided and, where safe, autonomous execution** across offer operations (CPQ/catalog), customer care journeys, and billing/assurance workflows.

● In this whitepaper

we present the reference architecture and agent patterns behind Agentic BSS, describe how it is realized across **Veris Suite** (carrier-grade BSS) and **Veris Agent Builder**, and walk through two concrete end-to-end scenarios: a CSR new-connection journey and broadband delay diagnosis/rescheduling. We also outline the evaluation methodology, operational controls, and deployment roadmap required to take agents from demo to production.

Contents



01 Executive Summary

02 Why Agentic BSS Now

03 Defining Agentic BSS

04 Why It Matters & Business Value

05 The Architecture of Agentic BSS

06 AI-Assisted Order Activation: Hero Flow
Example (New Mobile Connection with
Port-In)

07 Deployment and Integration
Considerations

08 Risks, Limits, and Mitigations

09 Evaluation and Measurement
Framework

10 Implementing Agentic BSS in
Cloudnet.ai's Product Stack

11 Competitive Landscape

12 Conclusion & Appendices

01

Executive Summary



Definition

Agentic BSS is an AI-native operating model and execution layer

that transforms customer and operational intent into **governed, auditable actions** across telecom Business Support System (BSS) and Operations Support System (OSS) platforms.

It enables specialised AI agents to coordinate workflows—such as product configuration, order fulfilment, service assurance, billing, and customer care—through **controlled APIs and event-driven triggers**, while preserving the BSS/OSS as the **system of record** and enforcing telco-grade controls (policy checks, human approvals where required, traceability, and rollback).

Why now

Telecom operators are being pushed into higher operational complexity at the same time as economic headwinds increase pressure to improve productivity.

PwC's Global Telecom Outlook 2024—2028 characterises the sector as

growing—slowly

Projecting global service revenue growth of ~2.9% CAGR through 2028 (below inflation), while operators still face a continual need to invest in infrastructure—tightening the case for cost-to-serve reduction and faster monetisation of new services.

Meanwhile, industry architecture direction is increasingly explicit that operators must modernise for **automation at scale**: TM Forum's Open Digital Architecture (ODA) positions a blueprint for simplifying enterprise architectures, modernising for a plug-and-play ecosystem, and automating IT and network operations.

McKinsey & Company

In parallel, telecom executives are prioritising agentic AI as the next frontier:

McKinsey reports that nearly 42% of telco executives identify scaling “agentic use cases” across functions as a priority for 2025 (notably in customer service).

Gartner describes agentic AI as a shift from systems that primarily generate or summarise information to systems that can **act autonomously to complete tasks**, and forecasts materially higher levels of automation in service operations over the coming years.

Taken together, the pressures are structural: rising service complexity (5G/IoT, enterprise SLAs, omnichannel expectations) meets persistent cost constraints, while traditional automation (rules, RPA, and assistant-only copilots) hits diminishing returns in exception-heavy, fragmented, multi-system environments. Agentic BSS addresses this gap by embedding AI into the **execution fabric** of BSS workflows—without bypassing governance.

The solution —

Cloudnet.ai's Agentic BSS stack

Cloudnet.ai implements Agentic BSS as a layered architecture that separates control from execution, enabling scalable automation while keeping telco-grade governance explicit.



Veris Suite

(execution plane / system of record)

The carrier-grade BSS/OSS backbone for CRM, product and service lifecycle, order management, billing/charging, and assurance. Veris Suite remains authoritative for transactions and compliance-sensitive state, exposing capabilities through secure APIs and event streams. This aligns with industry moves toward standardised, event-capable integration surfaces that enable intent-driven automation while preserving interoperability.

Veris Agent Builder

(control plane / agent orchestration)

The AI agent runtime and orchestration layer that converts intent to structured requirements, routes to the correct playbook, coordinates multi-agent execution, and invokes tool adapters (APIs/events) under bounded autonomy. Veris Agent Builder is designed to manage exceptions and maintain execution state across multi-step journeys while producing deterministic artefacts for validation, audit, and replay.

Order Genius Hub

(control surface / user interaction)

The user-facing copilot experience embedded in CRM workflows. It captures intent, guides structured field completion, renders recommendations as actionable items, and supports draft-firstwrite-back so humans can review and approve sensitive actions before commit.



Measurable outcomes — value waterfall from mechanisms to telco KPIs

Agentic BSS is designed to translate architectural change into operator KPIs by reducing preventable friction (system switching), preventing avoidable exceptions (structured capture/validation), and closing loops faster (event-driven remediation).



Value waterfall:

mechanism → KPI impact → business benefit

1

Structured data capture and validation at the point of intent



lower order fallout rate and less rework



faster order completion, improved revenue realisation, and lower cost-to-serve

(In telecom fulfilment operations, “fallout” commonly refers to orders that enter an exception state during orchestration and require user intervention.)

3

Cross-domain execution across BSS/OSS tools (catalog → eligibility → order → provisioning → billing)



higher first contact resolution (FCR)



fewer repeat contacts and escalations, improving customer experience while reducing contact volume

(FCR measures the percentage of contacts resolved on the first try and is closely linked to efficiency and cost.)

2

Reduced system switching through guided orchestration



lower average handling time (AHT)



higher CSR capacity and improved operational efficiency

(AHT is widely used as a contact-centre metric for the time required to complete a transaction, including talk/hold/wrap work.)

4

Closed-loop remediation (event triggers + diagnostics + playbooks)



reduced backlog and fewer manual interventions



improved SLA performance and lower operational risk as exceptions are resolved sooner and more consistently



Governance and trust posture

Agentic BSS is only deployable at scale when governance is part of the execution design, not an afterthought. Cloudnet.ai's approach emphasises **draft-first execution**, with bounded autonomy, explicit policy gates, and full traceability:

Draft → **validate** → **approve** → **commit**

as the default progression for customer-impacting or high-risk actions.

Human-in-the-loop approvals

where policy, commercial risk, or regulatory exposure demands oversight; constrained auto-execution where risk is low and controls are satisfied.

Auditability and replay:

session traces, tool-call logs (inputs/outputs/timestamps), decision artefacts, and outcome records are produced to support incident review, compliance reporting, and continuous improvement.

Policy enforcement and explainability:

deterministic checks remain separate from model reasoning; recommendations and actions include rationale, constraints applied, and uncertainty signals.

This posture is consistent with widely adopted AI risk management guidance that stresses lifecycle controls, documentation, monitoring, and governance for trustworthy AI use in operational environments.

This whitepaper is intended for senior telecom stakeholders evaluating governed autonomy in the BSS domain:

CIOs/CTOs and enterprise architects

modernising BSS/OSS architectures for scalable automation and measurable outcomes.

1

BSS/IT owners and operations leaders

accountable for cost-to-serve, order integrity, resolution performance, and change velocity.

2

Enterprise product owners

delivering complex, SLA-backed offers that require faster, more reliable fulfilment across domains.

3

Industry analysts and transformation leaders

assessing whether “agentic” approaches translate into credible architecture, governance, and KPI impact at production scale.

4

Target audience



Why Agentic BSS Now

02



Telecom operators today face unprecedented complexity

in their Business Support Systems (BSS).

The rise of **5G, IoT, converged services, and demanding enterprise SLAs** has multiplied the number of products, plans, and real-time parameters that BSS must handle.

At the same time, customers expect **seamless omnichannel experiences** — moving fluidly between app, web, and human channels — which traditional BSS architectures struggle to orchestrate end-to-end. The result is operational fragmentation and cost pressure.

Customer service representatives (CSRs) often act as

”

operational glue

manually stitching together processes across siloed tools and screens.

All of this drives up cost-to-serve and increases the chance of errors — for example, high order fallout risk when a small inconsistency causes a customer order to fail partway and require costly rework.

In short, legacy BSS workflows are **labor-intensive, error-prone, and unable to keep up** with the complexity and speed of modern telecom services. **CSPs (Communication Service Providers) find themselves in a bind: revenue from traditional services is flat, yet operational expenses keep rising, making efficiency gains a top priority.**

This leads to

inefficiencies (time wasted switching between systems)

tool sprawl with duplicate data entry

and even **manual policy enforcement** (human checks for eligibility, compliance, etc.)

...



Telcos have tried to address these pain points with traditional automation—

hard-coded

business rules, RPA scripts, and basic AI chatbots. These measures have delivered some incremental improvements, but they fall short of what's needed.

Rules and RPA bots can automate routine tasks, but they are brittle when facing new scenarios or exceptions. Basic chatbots can answer simple queries, but they can't truly execute complex actions across BSS/OSS systems.

In fact

most operators remain stuck in

”

islands of automation

where each tool or channel is optimized in isolation.

The limitations of this approach are becoming apparent in the 5G era:

as one analysis noted, telecoms made progress with scripts and bots, “digitising and optimising their operations” incrementally, “but as we move deeper into the 5G... era... the limitations of automation have become apparent.”

In other words, simply speeding up siloed tasks is not enough when the underlying processes themselves are fragmented.

The tougher problems

like coordinating a multi-step order across billing, provisioning, and support, or diagnosing a network issue that spans domains — **still require human intervention**. CSRs and ops teams remain the fallback, constantly handling edge cases and bridging gaps between systems. This not only erodes the efficiency gains but also highlights a structural limitation: **conventional automation doesn't fundamentally change how the BSS operates; it just papers over cracks with point solutions.**



Even the recent wave of AI assistants and “copilots” has not fully bridged this structural gap.

Conversational AI can engage customers, and generative AI (like LLMs) can draft responses or summarize information, but on their own **they can’t remake the underlying BSS architecture.**

An AI chatbot layered on top of legacy systems is still constrained by those systems’ silos and lack of real-time decision automation. Indeed, standalone LLM-based assistants lack telecom-specific knowledge and direct access to live BSS data, limiting their usefulness for complex operations. They might talk to a customer in natural language,

but they aren’t inherently equipped to perform multi-step transactions, enforce policies, or orchestrate tool calls across multiple platforms.

In practice

Whenever a query goes beyond a bot’s script, it is handed off to a human who must interpret the request and execute it using various systems — the same old human glue. What’s needed is more than an assistant on the side; it’s a **system-level shift** in how the BSS functions.

This is the core idea behind Agentic BSS.

Rather than treating AI as an add-on, Agentic BSS embeds intelligence **throughout the BSS lifecycle**. Autonomous AI agents are woven into core processes — product catalog management, CPQ, order orchestration, billing, customer care, assurance, and more — acting on behalf of users and operators. These agents are **goal-directed** and context-aware, not just answering questions but actually making decisions and taking actions within governed bounds.

For example

An agent in an order management role could detect an order issue and automatically invoke the necessary OSS/BSS tools (“**tool call orchestration**”) to resolve it, instead of waiting for a human to troubleshoot. Crucially, this is done with “**draft mode**” governance — agents prepare changes or recommendations in a reversible, reviewable draft state whenever appropriate, ensuring **human oversight and policy compliance** before final execution.

This approach maintains **telecom-grade trust** (no wild unchecked AI behavior) while still radically speeding up operations. In essence, Agentic BSS aims to **replace the human middleware** with intelligent middleware. The AI agents become the new operational glue — but unlike humans, they operate at digital speed, 24/7, and according to consistent rules.

Why is now the right moment for this AI-native paradigm?

A convergence of technological and market factors has made **Agentic BSS** **feasible and urgent**. On the technology side, recent advances in AI and software architecture have provided the building blocks:

- large language models (LLMs) with deep reasoning capabilities
- scalable agent frameworks
- event-driven microservice architectures
- robust APIs for everything in the BSS

These enable systems that can understand unstructured inputs, interact with multiple services, and learn from outcomes. In fact, industry experts note that capabilities like autonomous order fulfillment and proactive care are “**becoming technically viable thanks to advancements in LLMs, multi-agent orchestration, and event-driven architectures.**”

On the market side

Nearly
42%



The pressure to transform has never been greater. Telecom executives are increasingly prioritizing AI at the core of their business.

Nearly **42% of operators’ C-level leaders say that scaling “agentic AI” use cases is a top priority for 2025** (especially in customer service), reflecting a broad recognition that AI-driven autonomy is the next frontier.

Moreover

Telecom standards and ecosystems are catching up — open APIs (TM Forum, etc.), cloud-native BSS components, and AI governance frameworks are maturing, reducing the integration barriers that existed just a few years ago. In short, **the technology is ready and the need is acute**. We have the ingredients (powerful AI engines, APIs, real-time data pipelines) and a clear mandate to dramatically improve agility and cost-efficiency.

Agentic BSS is about seizing this moment to leap ahead, moving beyond “AI-powered” patchwork to an AI-native BSS architecture that can handle the complexity of modern telecom operations autonomously. By doing so, operators can finally break free from the slow, reactive mode of the past and enter an era of proactive, self-optimizing operations — all while staying in control through proper governance. This is the paradigm shift Cloudnet.ai is championing at exactly the right time.

03

Defining Agentic BSS



Traditional telecom **Business Support Systems (BSS)** have been largely passive record-keeping platforms, often requiring human operators to manually drive complex processes across multiple modules and systems.

This leads to

multi-step wizards

overflowing forms

...

error-prone manual checks

CSRs (customer service reps) juggle separate BSS, OSS, and ticketing systems

In contrast

An **Agentic BSS** is an AI-native BSS designed from the ground up to embed autonomous intelligent agents throughout the BSS lifecycle.

The goal is to transform the CSR from a “form-filling machine” into a decision-maker, by offloading complex backend coordination to AI agents.

In essence

The BSS evolves from a passive system-of-record into an active, intelligent, and adaptive platform that can act, decide, and optimize with minimal human input.

This new model is often described as enabling

AI-to-AI operations”

— meaning processes are handled through AI agents interacting with each other and the environment, rather than through constant human-to-machine commands.



To us

Agentic BSS is formally defined as a carrier-grade BSS in which autonomous agents are deeply integrated to run core business processes end-to-end. Unlike legacy approaches that bolt AI onto the side of existing workflows, an Agentic BSS integrates intelligence natively into its architecture. The system can interpret high-level objectives, make context-aware decisions, and execute tasks in real time, all while ensuring outcomes remain **telco-grade** — auditable, compliant, and reliable.

In practical terms

Cloudnet.ai's Agentic BSS approach spans the entire BSS spectrum (from product catalog and CPQ to order management, billing, customer care, and assurance) with embedded AI capabilities. By combining telecom-grade trust with AI autonomy, this paradigm promises **radically faster rollout, lower operational costs, and richer customer experiences.**





Key Capabilities of an Agentic BSS:

Cloudnet.ai's internal design of Agentic BSS outlines several core capabilities that illustrate how it differs from a traditional system :

1

Natural Language—Driven Service Fulfillment:

A human user (e.g. CSR) can simply describe a customer's need in natural language, and the system's "copilot" interface will interpret the intent, gather the key information (e.g. phone number to port, plan preferences, device choice), perform validations, and automatically fill out the appropriate BSS forms. This turns lengthy form-based workflows into a guided conversational experience, drastically simplifying the user experience.

2

Multi-Agent Process Orchestration:

Under the hood, multiple specialized AI agents collaborate to handle complex logic in parallel.

For example, during a new mobile service signup, one agent checks number portability eligibility, another performs a credit check, a third queries device inventory, while yet another suggests the best plan/package.

A **supervisor agent** orchestrates these sub-agents and aggregates results, so the overall process completes in real-time. The agents access BSS/OSS functions through a unified tool layer (standard APIs), ensuring they operate within approved system boundaries.



3

Closed-Loop Diagnostics and Resolution:

The Agentic BSS doesn't just assist with new orders — it can also monitor and optimize active processes.

For instance, if a broadband installation order is delayed, an agent can automatically diagnose the cause (e.g. resource shortage or scheduling conflict) and attempt remedial actions. It might proactively propose a new installation time to the customer and update the order, or if needed, open a trouble ticket with all relevant details — all without manual intervention.

This closed-loop operation ensures issues are identified and addressed by the system in a timely manner, improving reliability and customer satisfaction.

4

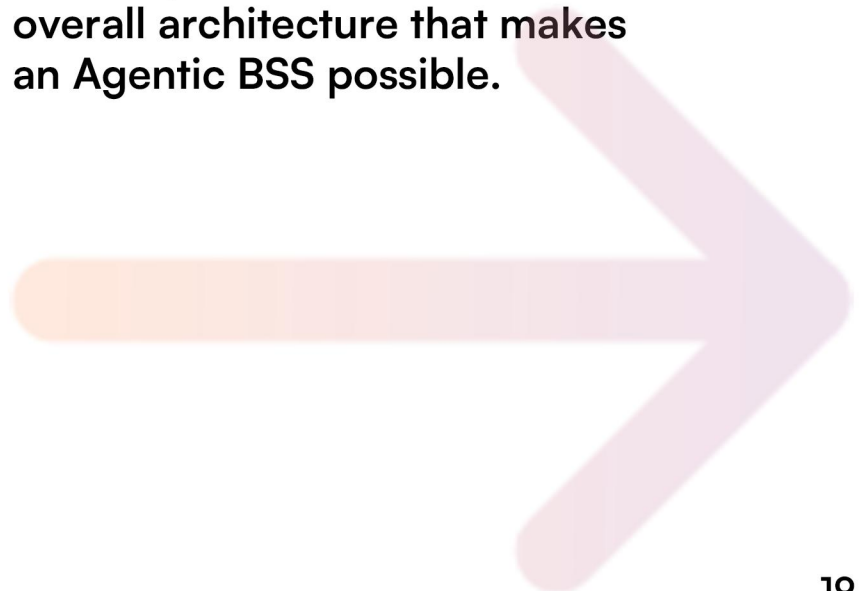
Scenario Playbooks & Configurability:

Each business scenario (order type, support case, etc.) in an Agentic BSS is defined by a **Playbook** rather than hard-coded logic. A playbook declaratively specifies the intent to handle, required input “slots” (fields), which agents should be invoked, the sequence of tool/API calls, and how the UI should present information. New or modified processes can thus be introduced by updating playbook configurations instead of writing code. This makes the BSS highly adaptable — operators can configure and version these AI-driven workflows and even do canary (gradual) rollouts of new automations. In short, the Agentic BSS is policy-driven and extensible, allowing telecom business processes to evolve much more rapidly than before.



By natively weaving these capabilities into the BSS, an Agentic BSS embodies a shift to a “self-driving” mode of operations. Routine tasks are handled autonomously and proactively, while humans provide oversight and handle exceptions.

Crucially, this is achieved without sacrificing the stringent requirements of telecom software — every agent action is auditable and every decision can be traced. The next sections describe how Cloudnet.ai realizes this vision through the **Veris Suite** and **Veris Agent Builder** platforms, and the overall architecture that makes an Agentic BSS possible.



Why It Matters & Business Value

04

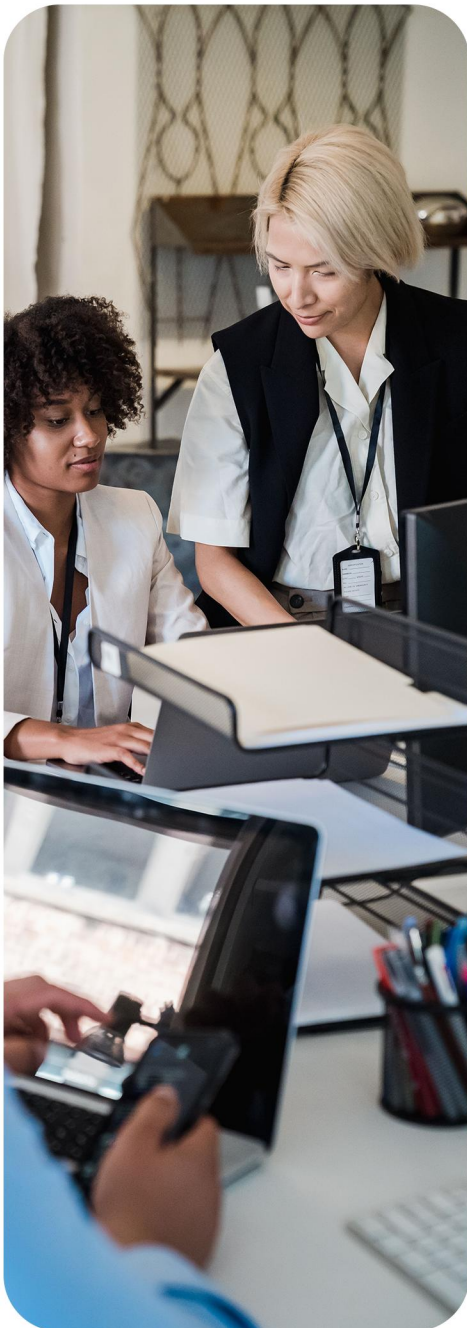


Ultimately, the Agentic BSS approach isn't just a technical novelty — it drives **tangible business outcomes** that telecom executives care about.

By transforming BSS processes from manual or semi-automated to truly autonomous, Cloudnet.ai's Agentic BSS delivers improvements that cascade through key operational metrics.

Think of it as a **value waterfall**:

each layer of efficiency gained flows into cost savings and better customer experiences downstream. The major impact areas can be summarized as follows:



● **Lower Cost-to-Serve:**

By offloading routine work from humans to AI agents, Agentic BSS **reduces the human effort per transaction** dramatically. CSRs spend less time on data lookup, form-filling, and cross-checking, which translates to fewer staffing hours per customer interaction. This directly lowers operational costs

Industry examples show the potential scale of savings — for instance, telcos that deployed AI-driven service bots saw cost per customer contact drop by 30—35%.

↓ **30—35%**

These reductions come from multiple sources: automated agents handle a large chunk of cases end-to-end (at near-zero marginal cost), and even when humans are involved, the AI augments them to work faster (e.g. prepping information, handling after-call tasks). In an environment where **margins are tight and traditional revenues have plateaued**, cutting the cost-to-serve by double digits is a critical lever. Agentic BSS attacks the labor overhead in BSS processes, allowing operators to scale support for new services without a linear increase in headcount . In short, it offers a step-change in productivity — doing more with the same or fewer resources.



Reduced Average Handling Time (AHT):

For customer-facing scenarios, Agentic BSS significantly **shrinks the time to complete an interaction or order**. Agents (the AI ones) work in parallel and at machine speed — fetching data, running diagnostics, and executing transactions in seconds. This means that when a CSR or customer is engaged in a task (like activating a service or troubleshooting an issue), the back-end steps happen faster, and often concurrently, rather than serially with waiting times.

Real-world pilots bear this out: in one case, using AI to summarize and pre-fill info from customer conversations cut **handling time by over 5 minutes per call**.

Multiply that across thousands of daily calls and you not only serve customers faster, but also free up agent capacity. Automated guided flows (driven by AI) also prevent the pauses and errors that extend AHT-

for example, the system won't pause to search for information or stumble on a missing field, as it proactively gathers needed context. Every minute shaved off transactions is a win for efficiency and customer satisfaction.

Customers spend less time on hold or repeating information, and CSRs can resolve more issues in a shift. **Shorter AHT** directly contributes to lower costs and higher customer satisfaction scores.

Higher First-Call Resolution (FCR):

Perhaps one of the most visible benefits is the jump in **first-contact resolution rates**. An Agentic BSS enables issues to be fully resolved in one interaction far more often, thanks to its ability to **orchestrate multiple tools and parse all required inputs (“slot-parsing”) on the fly**.

For example, if a customer calls with a billing dispute that actually involves a mix of billing, provisioning, and product catalog data, a traditional setup might require multiple teams or call-backs.

In contrast, an autonomous BSS agent can pull the necessary details from all relevant systems, perform corrections or adjustments, and present a solution within that first call. It ensures **no key information slips through** — using NLU to extract entities like dates, plan IDs, or error codes from the conversation — so that by the end of the call, nothing is pending external follow-up.

The results can be dramatic: one major operator's AI help desk bot improved customer issue resolution rates by about **60%** (i.e. far more customers got their issue fully resolved on the first try).

↑
60%

Higher FCR means fewer repeated calls or escalations, which again lowers total effort and improves customer experience. It also boosts customer trust — subscribers feel the company is competent when their problem is solved without being passed around. By **coordinating all the “slots” of information and all the necessary actions in one go**, Agentic BSS turns what used to be multi-step, multi-day cases into one-touch resolutions.



Fewer Order Fallouts and Errors:

In the traditional BSS, **order fallout** — orders that fail to complete properly and require manual intervention — is a persistent headache. It might be caused by a missing field, a policy check that wasn't done upfront, or a downstream system mismatch. These fallouts are not only costly (each requires a human to investigate and fix) but also delay revenue and annoy customers. Agentic BSS attacks this problem at its root. Because autonomous agents handle orders in a **“draft-then-commit” mode with validation**, they catch issues early.

For instance, an agent preparing a new service order will automatically verify that all required fields are present and consistent, perhaps even querying an external system if something looks off, before final submission.

If a step does fail, the agent can immediately try alternate paths — what we call **self-healing order fulfillment**.

An example from a trial: when a service activation failed due to a resource conflict, the agent dynamically rerouted the workflow (rescheduling the installation to an available time and auto-creating a support ticket for the delay) **without any manual intervention**, thus preventing a fallout.

This kind of autonomy **greatly reduces fallout rates**, ensuring orders complete in one pass or are intelligently corrected in-flight. The net effect is fewer lost orders, faster revenue realization, and far less “clean-up” work for operations teams. In addition, by consistently enforcing business rules and filling information accurately, the system **reduces errors across the board**, from billing mistakes to configuration faults. All of these translate to real money saved and a smoother customer journey (e.g., a customer's new device is activated on time with no hitches, because the AI handled all prerequisites and checks in the background).

These improvements are not hypothetical — they are grounded in measurable outcomes, and they **stack together**. Lower cost-to-serve, shorter AHT, higher FCR, and fewer fallouts collectively yield a powerful business case: better customer satisfaction and loyalty on one hand, and significantly improved operational efficiency and agility on the other.

A recent Gartner prediction, for example, foresees that by 2029 agentic AI could autonomously handle 80% of common service issues, potentially cutting operating costs by 30%. Agentic BSS is the pathway to realize those kinds of gains in practice.

handle
80%
of common
service issues

cutting operating
costs by
30%



How are these gains achieved?

Cloudnet.ai's approach brings a few key **mechanisms** to bear, based on real implementation experience:

● **Unified, context-rich workspace (reduced system switching):**

Agentic BSS provides a unified interface or “copilot” workspace that brings relevant information and actions to the CSR (or to the AI agent itself) without constant context switching. Instead of Alt-tabbing through CRM, billing, order management, and network systems, the human or AI agent works through one orchestrated canvas. This dramatically reduces the time lost in navigating disparate tools. It also means all context is in one place — fewer things get missed.

For example, when handling a broadband installation issue, the system can automatically pull customer info, order status, and network resource data into one view. The CSR isn't searching three different systems; the **Agentic platform acts as the aggregator**, so decisions are faster and more informed.



Automated diagnostics and action orchestration:

When problems arise, Agentic BSS employs **embedded intelligence to diagnose and fix issues automatically**. Instead of a human manually querying various systems for clues, an AI agent proactively calls the relevant APIs/tools to pinpoint the root cause.

For instance, if an order is delayed, an **OrderAgent** might call the order management API to get real-time status, then an **OSSAgent** to check field service schedules or resource availability. Within seconds, the agent can determine that “Technician resource conflict — not enough technicians for the original time slot” is the root cause of a delayed installation. It can then suggest next-best actions (e.g. reschedule the appointment, or create a trouble ticket to escalate) to the CSR or **even execute them directly**.

This kind of **tool call orchestration** — automatically invoking the right sequence of operations across BSS/OSS — turns what used to take a trained human many minutes (or require coordination across departments) into a near-instant, single-step outcome. In essence, the AI agent does the heavy lifting of **interpreting data and executing the remedy**, while the human can simply confirm or supervise. The benefit is faster resolution and a huge relief on front-line staff workload. It’s the difference between knowing there’s an issue and actually fixing it in real time, without waiting.

Structured data capture and “draft mode” governance:

A subtle but powerful aspect of Agentic BSS is how it **captures and uses data in structured form** to drive processes. Rather than relying on a human to mentally compile information or fill forms from memory, the AI agents use techniques like **slot parsing** to extract key inputs from unstructured sources (natural language requests, past records, etc.) and populate the necessary fields automatically.

For example, if a CSR says in chat, “The customer wants to upgrade to the 50GB plan starting next billing cycle,” the system can parse out the target plan, the effective date, the customer’s ID, etc., and **generate a draft order** with those specifics filled in. This draft is then presented for review/approval — a form of **draft mode governance** that ensures the CSR or a supervisor can validate any AI-proposed transaction before it’s finalized.

By operating in draft mode, the platform maintains strict **compliance and control**: every AI-initiated change can be audited and must pass business rules (no surprise changes go live without oversight). Yet from the user’s perspective, the effort is minimal — the heavy work (filling out all the fields, checking consistency) is done by the AI. This mechanism both **boosts accuracy and saves time**. It virtually eliminates those scenarios where an order falls out because a field was missed or a detail was miscopied. Furthermore, it **empowers human staff to focus on decision-making** and customer interaction, rather than being “form fill robots”. In turn, this contributes to higher quality service and job satisfaction for employees (who can trust the system to handle grunt work and not worry about remembering every rule). The structured data approach also means that the BSS continuously accumulates clean, machine-readable data from every interaction, which feeds back into analytics and machine learning to further improve operations.



Finally, it's worth addressing the **strategic gap** that Agentic BSS fills in the telecom industry.

Even as operators have invested in digital channels, self-service apps, and AI chatbots, the truth is that **most still rely on humans to bridge the last mile of complex service fulfillment.**

A web portal might let a customer order a new service, but if anything unusual happens (say the customer has a mix of discounts, or a network glitch occurs), a human in the back office scrambles to reconcile it.

Similarly, a basic AI bot might answer FAQs, but a nuanced issue (like a combined billing and network problem) gets escalated to a live agent who then navigates legacy systems to resolve it.

In essence, the industry has improved the surface (the digital front-ends) but the core processes often remain manual or disconnected — **human “middlemen” are still holding systems and data together.**



As one industry commentary put it

the bottleneck isn't people's skill, but that “the core issue lies not with the people but with the processes they are forced to use” — fragmented processes that “**AI is uniquely positioned to address at scale.**”

This is exactly why Agentic BSS matters. It tackles that long-standing fragmentation by redesigning BSS operations to be AI-driven and integrated from the ground up. It closes the gap between what digital channels promise and what legacy BSS actually deliver.



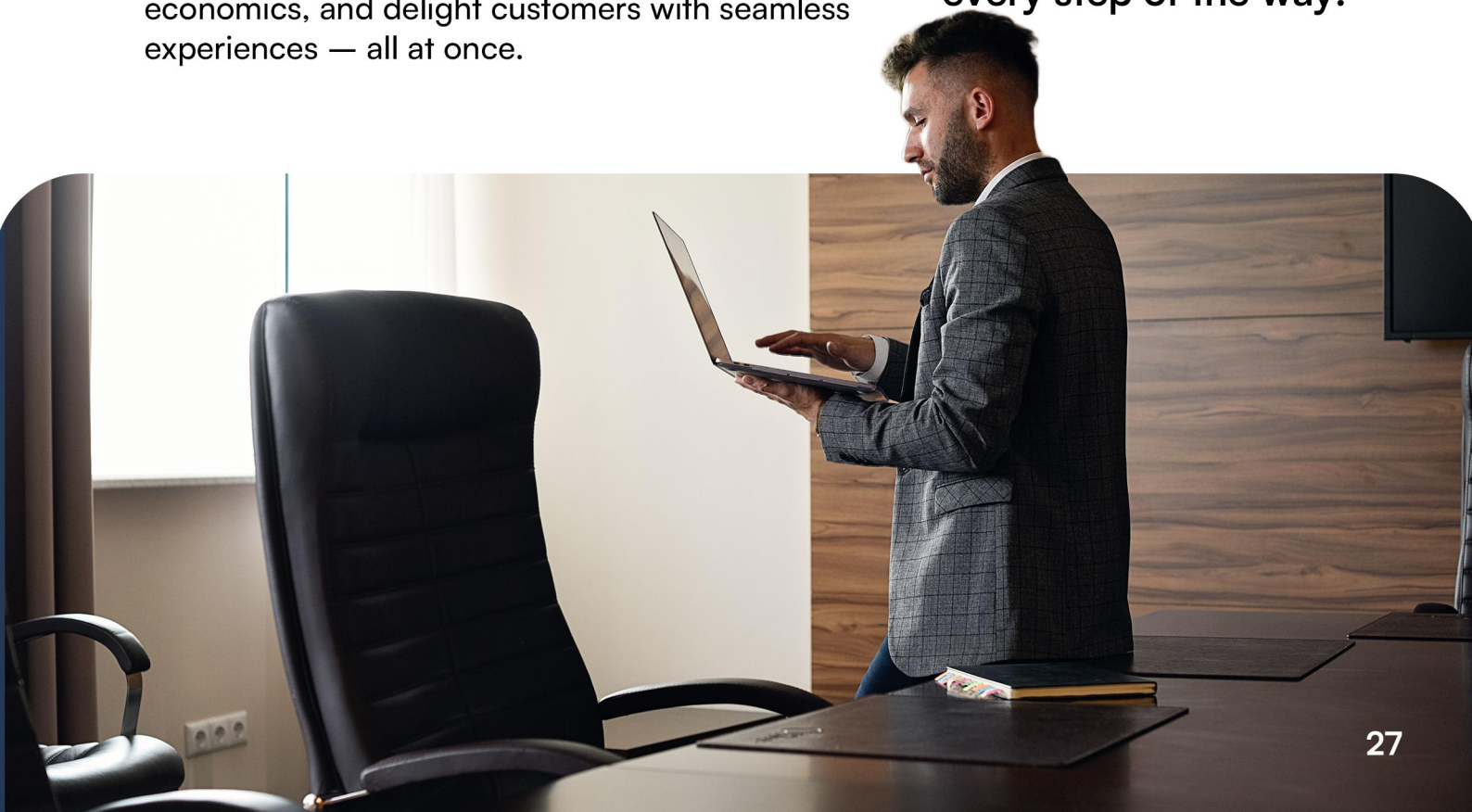
For a telco CIO or CTO, this means finally realizing the vision of a **truly digital, AI-enabled operator**: one where systems don't just pass data to humans to interpret, but systems **understand and act on data themselves** in alignment with business goals.

The business value is not only in the specific metrics (cost or time saved) but in the strategic agility it unlocks. Agentic BSS allows operators to launch new offerings faster (since AI agents can adapt processes quickly), to assure quality with less overhead, and to deliver a level of proactive customer care that differentiates them in a competitive market.

It's a transformation from a **passive, reactionary BSS** into an **active, intelligent BSS**

that can truly “run the business” in partnership with its human stewards. This paradigm shift — from AI as a bolt-on tool to AI as an **embedded, autonomous workforce within BSS** — is what will define the next generation of winners in telecom. Cloudnet.ai's Agentic BSS is designed to be the enabler of that shift, marrying telecom-grade reliability and governance with cutting-edge AI autonomy. The result is an operator that can move at digital speed, drastically improve its economics, and delight customers with seamless experiences — all at once.

This is **why Agentic BSS matters**: it's not hype or ivory-tower theory, but a pragmatic re-imagining of BSS to meet the demands of the 2025+ telecom landscape, delivering real value every step of the way.





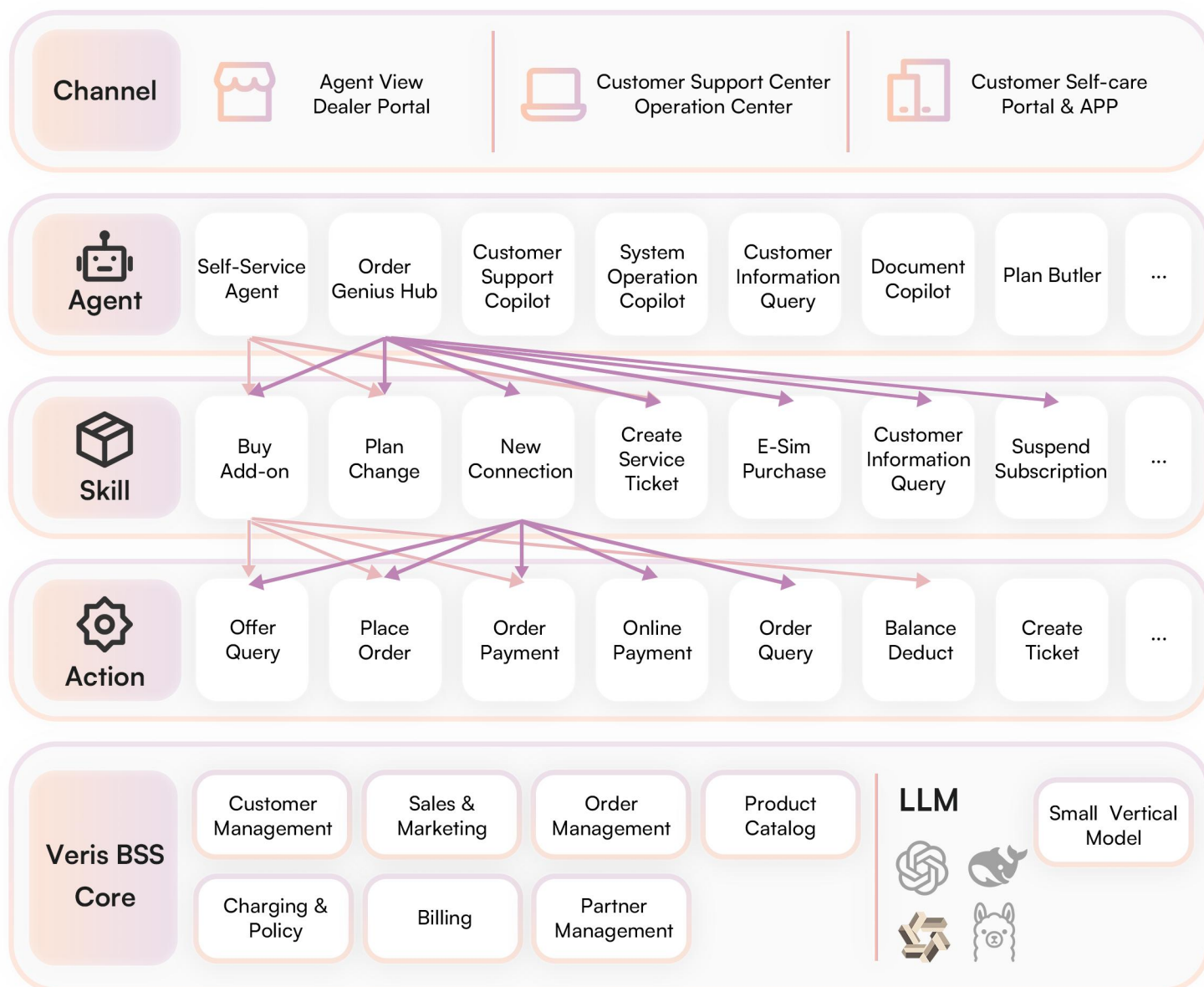
05

The
Architecture
of Agentic
BSS



Veris Agentic BSS

-Multi-AgentSystembasedonamulti-modallarge-languagemodel



Overview

The architecture of an Agentic BSS combines a modern AI orchestration layer with the robust transaction processing of a traditional BSS. At a high level, we can think of the Agentic BSS in three parts:

- (1) a user-facing “copilot” interface for natural language interactions,
- (2) an AI-driven agent orchestration core
- (3) the underlying BSS/OSS systems of record.

In Cloudnet.ai’s implementation, these correspond to **Order Genius Hub**, **Veris Agent Builder**, and **Veris Suite** respectively. The Order Genius Hub serves as the unified entry point for users (e.g. CSR agents), while the Agentic BSS runtime in Veris Agent Builder acts as the “central brain” coordinating all intelligent behavior.



The existing BSS/OSS modules (Veris Suite and connected operational systems) function as the execution arm — they are the stable, compliant back-end that actually fulfills orders, updates accounts, triggers network actions, etc., at the direction of the AI layer.

From a technical perspective, the solution is implemented via a combination of **multiple agents**, a **tool/API abstraction layer**, **playbook-defined workflows**, and a **draft-mode UI integration** for human oversight.

These elements work together as follows:

1

Veris Suite

(System-of-Record Layer):

Veris Suite is Cloudnet.ai's carrier-grade BSS/OSS backbone — encompassing modules for CRM, order management, billing, etc. — now extended with AI-friendly integration points. All core business logic and data (product catalog, customer data, order processing, trouble ticketing, etc.) lives here.

Veris Suite exposes **open APIs and event hooks** for external agents to query or modify data in a controlled manner. In other words, it provides the “nerves and muscles” that the AI agents will move. Crucially, by using official APIs (aligned with TM Forum Open API standards), the agent layer never bypasses governance: every action goes through the same validations and audit trails as a manual operation.

This design preserves telco-grade compliance — for example, if an agent wants to create an order or adjust a plan, it must call the same service that a human CSR would, ensuring all existing business rules apply.



2

Veris Agent Builder

(AI Orchestration and Agent Layer):

Veris Agent Builder is the AI-native platform that builds and runs the autonomous agents. It can be thought of as a low-code **agent builder** and runtime, where developers define **Actions**, aggregate them into higher-level **Skills**, and package those into intelligent **Agents**. At runtime, Veris Agent Builder hosts the **Agentic BSS Runtime** — the multi-agent brain of the system.

This includes a top-level **Supervisor (orchestrator) agent** and an array of **domain-specific agents** for particular tasks (e.g. a PortabilityAgent for number port-in checks, an OfferAgent for recommending plans, a DeviceAgent for inventory lookup, an OrderAgent, TroubleTicketAgent, etc.). The Supervisor (internally codenamed “DeepAgent”) is responsible for planning and coordinating workflows: it interprets the user’s request or situation, breaks the problem into subtasks, and assigns those to the appropriate domain agents.

Each domain agent is equipped with the logic (or ML model) to handle its specialized task and knows how to call the relevant backend APIs via the tool adapter. Veris Agent Builder provides a **Tool Adapter layer** that abstracts BSS/OSS APIs into a consistent interface for agents.

For example, an agent doesn’t need to know the details of SOAP vs REST or a specific database call — it will call a high-level “CheckEligibility” tool, which under the hood invokes the correct Veris Suite API.

All agents run concurrently when possible, enabling parallel execution of tasks that would traditionally be sequential. This dramatically speeds up complex transactions. The Veris Agent Builder layer also manages the lifecycle of playbooks (loading the right playbook for the scenario) and handles prompt generation for any large language model (LLM) queries if used (e.g., formulating the recommendations query to an LLM).



3

Order Genius Hub

(User Interaction Layer):

The **Order Genius Hub** is the AI-assisted user interface, essentially a CRM Copilot experience built into the operator's CRM front-end (Veris CRM). It appears as a chat-based assistant alongside or within the CRM screens, guiding the CSR through each interaction. Instead of navigating many pages, the CSR can converse with the system.

For instance, the agent can ask

“

What does the customer need today?

and the CSR might type

”

They want to activate a new mobile line and port their number

The Copilot (Order Genius Hub) will recognize this intent and automatically bring up the unified order interface for a new mobile connection.

The UI is composed of smart components — such as a “**Key Information Card**” that pops up to collect minimal required data for the request, and context-sensitive drawers or forms for any additional details. The Copilot interface pre-fills what it can (using existing customer data and the understanding of the request) and asks for any missing info in a conversational manner. Once the CSR provides the inputs, the system triggers the backend agent orchestration. Throughout this process, the Order Genius Hub updates the CRM UI in real time-

for example, after analysis, it might display a set of **plan/device**

recommendations generated by the agents and underlying LLM intelligence.

The CSR can simply click on the recommended option, and the Copilot will then auto-fill the detailed order form in the BSS with the selected plan, device, number, etc.

All of this happens within the familiar CRM interface, but with far fewer clicks and manual entries. In effect, the Order Genius Hub turns complex multi-step transactions into an intuitive guided experience, acting as the front-end manifestation of the Agentic BSS's intelligence.



Workflow Example:

To illustrate how these components work together, consider a “**New Mobile Connection with Number Porting**” scenario (one of the Agentic BSS storylines).

A CSR initiates the process by typing a request into Order Genius Hub (e.g. “Activate a new 5G plan with number port-in and a device installment”). Immediately, an **NLU module** in the Copilot Core layer parses this input, identifying the intent as New Mobile Line Activation and extracting key entities (port-in request, device installment, etc.). The **Scenario Router** then selects the appropriate **Playbook** for this scenario — essentially a predefined workflow called “MobileNewActivation” (as configured by the telecom’s architects).

This playbook tells the system which agents to invoke and in what sequence. The Supervisor agent in Veris Agent Builder takes over, launching multiple domain agents in parallel according to the playbook: for example, a PortabilityAgent checks the number port-in eligibility with the external MNP system, a CreditAgent might run a credit score check via an API, an OfferAgent queries the product catalog to find suitable plans, a DeviceAgent checks device stock availability, and so on. Each agent uses the Tool Adapter to call into Veris Suite or external OSS/IT systems as needed, all in real time. Within seconds, the Supervisor gathers the results from these agents — say the portability check is approved, the credit check is clear, and the OfferAgent has compiled the top 3 recommended bundles (e.g. a 5G plan + device + insurance combo).



At this point, the Copilot UI (Order Genius Hub) presents the CSR with the recommendations in a friendly format — e.g. “**Recommended Bundle:** Unlimited 5G Plan + iPhone 15 on 24-month installment (monthly \$70) with AppleCare; **Alternatives:** [...]”. The CSR can review these options and confirm the selection. Once confirmed, the chosen configuration is automatically **applied to the order:** the Copilot fills out the order details on the standard BSS order page (the CSR can see the fields populate instantly). Any remaining non-critical fields (like default shipping method, notification preferences) are set to defaults to save time.

The complete order is now in Draft status, visible on the usual order summary screen for the CSR. The CSR then simply clicks the final **Submit** (just as they normally would), and the order goes through the usual BSS transaction submission, invoking all downstream systems as it would if the CSR had manually entered everything.

This “**draft then confirm**” **control mechanism** is important — it ensures that while the AI automates 99% of the process, the human still reviews and authorizes the final submission, and the transaction is recorded through the normal auditable BSS workflow. (Insert flow visualization here to illustrate the step-by-step orchestration).





The same architectural pattern applies to other use cases.

For example, in the case of a **stalled broadband installation**, an OrderDiagnosticsAgent could detect that an installation order has not progressed, trigger a network resource check via OSS (through a Tool), then an OSSAgent and SchedulingAgent might collaborate to find a new available time slot, and even create a new appointment.

If the issue can't be resolved automatically, the system can escalate by invoking a TroubleTicketAgent — which uses a playbook to create a ServiceNow ticket and attach all relevant info (order ID, diagnostics, customer details) for human follow-up.

Notably, the trouble ticket creation itself is defined as a separate playbook (“create_trouble_ticket”) that can be triggered by the outcome of the diagnosis.

In all cases, the Agentic BSS runtime **keeps a complete log of actions**: all state changes from agents (diagnosis results, decisions made, etc.) are written to the session context and to the customer's profile or history in the BSS . This means the operator retains full visibility into what the AI did on behalf of the user — supporting compliance and traceability.



Playbooks and Governance:

A cornerstone of the architecture is the use of **playbooks** to strike a balance between AI autonomy and operational control. Because each scenario's process flow is explicitly defined (yet easily configurable), the operator can enforce business rules and best practices within those definitions.

The AI agents will follow the playbook's structure when executing tasks, rather than acting entirely free-form. This mitigates the risk of unpredictable AI behavior by constraining agents to known-good pathways for critical processes.

However, within those pathways the agents have the freedom to optimize (e.g. try multiple solutions, pick best offers, etc.). Playbooks can be updated as policies or products change, without rebuilding the underlying systems. Moreover, **Veris Agent Builder's low-code agent builder** ensures that any new agent skills (new Actions or API calls, new decision logic) can be introduced methodically and tested in isolation before being included in a live playbook.

All agent-driven actions can be subjected to the same **testing and approval** as traditional software changes, providing a clear governance model even as the system learns and adapts.

Maintaining Telco-Grade Trust:

Despite the high level of automation, the Agentic BSS architecture retains strict control over compliance, security, and reliability.

First, no agent can bypass the BSS's built-in safeguards — every operation (order creation, update, etc.) is executed via Veris Suite services that enforce validation, role permissions, and logging.

Second, the “**draft mode**” approach (where the AI prepares transactions but a human or a governed process finalizes them) ensures a human-in-the-loop for ultimate decision authority on customer-affecting changes. This is especially important in early deployments to build trust in the AI's choices. Over time, certain agent actions might be allowed to auto-execute (for example, auto-rescheduling an appointment might be fully automated at midnight with an SMS to the customer). Even then, the actions would be logged and reversible if needed.

Third, the architecture logs every agent decision and the data behind it — providing an **audit trail** for compliance and for refining the AI. If an agent recommended a plan that was later disputed, the logs can show which inputs (customer profile, preferences, product rules) led to that recommendation. This transparency is crucial in regulated telecom environments.



Finally, because the Agentic BSS is built on

Cloudnet.ai's proven BSS platform, it inherits robust enterprise features: scalability to carrier-grade transaction volumes, high availability, and alignment with industry standards (for example, Open APIs and TM Forum frameworks).

The introduction of AI-driven capabilities does not compromise these fundamentals; instead, it enhances the BSS with a layer of intelligence. The result is an architecture that **marries AI innovation with carrier-grade operations** — allowing telecom operators to innovate at software speed without losing the rigor and control that their business demands.



This architecture sets the stage for Cloudnet.ai's demonstrations (**e.g. at MWC Barcelona 2026**) of “self-driving” BSS in action, where real-world scenarios like offer management, customer care, and billing assurance are handled largely by AI agents with minimal manual intervention, all under the governance umbrella of the Agentic BSS design.

The Agentic BSS is not a replacement of human expertise but a reallocation — letting humans focus on complex decision-making and customer engagement, while the AI agents handle the heavy lifting of process execution in the background. Together, Veris Suite and Veris Agent Builder (with the Order Genius Hub at the forefront) deliver a harmonious system that is both **intelligent and trusted**, fulfilling the promise of an AI-native BSS.

AI-Assisted Order Activation: Hero Flow Example

(New Mobile Connection with Port-In)

06



CSR Journey Narrative:

A customer calls to activate a new mobile line by porting in their number from another carrier. Instead of navigating a complex multi-page wizard, the CSR opens the **Order Genius Hub** Copilot interface embedded in the CRM. In natural language, the CSR enters the request:

“

Customer wants to port in a number from Vodafone, needs a 5G plan (~€40 budget), an iPhone 14 on installment, and device insurance.

The Copilot intelligently recognizes this as a new mobile service activation intent. It automatically navigates to the unified order screen and pops up a tailored

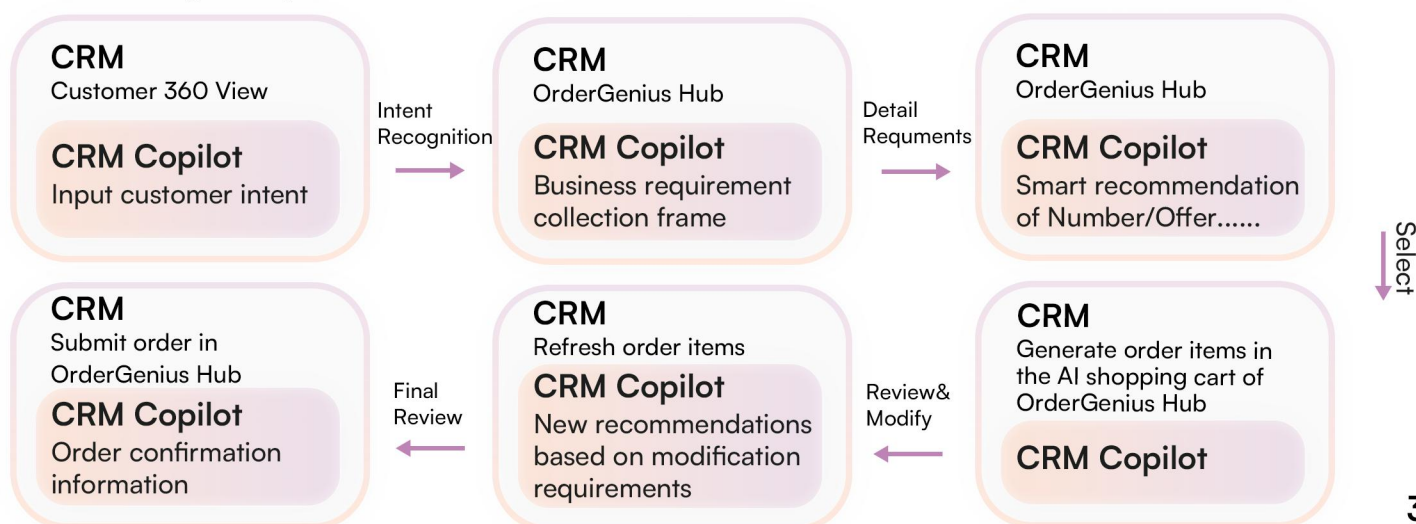
Key Order Info”

form pre-filled with details parsed from the CSR's description (e.g. the port-in flag, plan preferences, device model). The CSR verifies or fills any missing fields (such as the exact phone number to port, the PAC code, and desired port-in date) and clicks **Submit**.

Within seconds, the Copilot's AI engine analyzes the requirements and calls multiple backend agents in parallel. The CSR sees a set of **intelligent recommendations** returned: the best-matched 5G plan for the budget and usage, an installment device offer (iPhone 14) that meets the no-upfront preference, and a suitable insurance option, along with a couple of alternative plans/devices for comparison.

The CSR reviews these suggestions in a single consolidated view — like an AI-powered shopping cart. With one click, the CSR selects the recommended plan, device, and insurance combination that fits the customer's needs. The system then automatically **writes back** these selections into the official BSS order entry form on the left side of the CRM screen. The CSR sees that all the order fields (plan, device installment, port-in number details, insurance) are neatly filled in. After a quick review, the CSR submits the order through the normal BSS process.

The order is now created in the BSS with all the required data, but the CSR only had to input a few key pieces of information instead of manually traversing multiple screens. This hero flow illustrates how Agentic BSS turns a complex activation into a guided, conversational experience — reducing CSR effort and errors while ensuring compliance with existing BSS processes.





Agents and Orchestration

Behind the scenes, a **multi-agent system** coordinates the fulfillment of the request. A central **Supervisor Agent** (or Agent Orchestrator) oversees the process and breaks the order task into domain-specific subtasks, delegating each to an expert sub-agent.

For this mobile activation scenario, four key **worker agents** come into play in parallel:



a Portability Agent



Offer/Plan Agent



Device Agent



Insurance Agent

Each agent focuses on one aspect (number port-in eligibility, plan recommendation, device selection, insurance offering). The Supervisor (implemented via a LangGraph workflow planner) invokes these agents concurrently and waits for all their results, managing the sequence and any dependencies (e.g. the Offer Agent might use the Portability Agent's output if the port-in affects available plans).

This orchestrated design “splits a complex business request into small expert tasks, each handled by a specialized agent, then merges the results”. The Supervisor aggregates the agents' outputs into a cohesive set of recommendations before returning them to the Copilot UI.

Notably, agents interface with existing BSS/OSS systems through a unified tool adapter layer, ensuring they act on real data. The CSR is unaware of this multi-agent choreography — they simply experience a fast, unified response to their request.



Tool/API Calls and Integrations

Each agent uses **Tool APIs** to perform its function by querying underlying systems in real time.

For example, the **Portability Agent** calls a Number Portability API to check eligibility of the provided number/PAC for port-in (verifying the number can be transferred and the PAC code is valid).

Portability Agent → Number Portability API

The **Offer Agent** calls into the product catalog/CPQ (Offer API) to find available 5G plans, applying the customer's budget (~€40—50) and usage needs as filters.

Offer Agent → product catalog/CPQ (Offer API)

The **Device Agent** uses a Device API to query the device catalog and inventory for an iPhone 14 (128GB) and calculates installment pricing options that match the “no upfront” constraint.

Device Agent → Device API

The **Insurance Agent** invokes an Insurance product API to retrieve suitable device insurance plans (e.g. a standard coverage, avoiding the highest-cost premium option). All these tool calls are executed via a unified **Tool Adapter** layer, which standardizes how agents invoke external APIs and handle responses.

The tool layer encapsulates integration logic (e.g. REST endpoints, payloadschemas like /np/eligibility/check, /catalog/plans, /catalog/devices, etc.) so that the agents simply pass structured inputs and receive structured outputs.

This design abstracts the BSS/OSS specifics from the agents, allowing reuse and easier maintenance. If any external API is slow or unresponsive, timeouts (e.g. 800—1500ms) and graceful degradation policies are in place — for instance, if the device inventory call times out, the system might proceed with plan recommendations first or use cached device data to avoid slowing the CSR down.

By leveraging these tool APIs, the multi-agent Copilot can autonomously retrieve all necessary info and make decisions in real time, without manual CSR lookups in multiple systems.



What can I do for you?

Type a message...

Upload File

Send, Shift + New Line

HTML Mode

Docs Mode

PPT Mode

Table Mode

PDF Mode

SQL Mode

Data Objects and Workflow Dataflow

Throughout the flow, the Copilot backend passes around well-defined **data objects** to maintain a clear contract between steps.

When the CSR submits the Key Order Info form, the input is captured as a **KeyOrderInfo** JSON object — essentially a structured summary of the customer's requirements.

This object includes fields like

NumberIntent

(with details for porting the phone number, e.g. the MSISDN, donor carrier, PAC code, desired port date)

PlanIntent

(textual needs and budget for the plan)

DeviceIntent

(whether a device is needed and any preferences like model and payment)

InsuranceIntent

(preference for insurance)

Any cross-cutting constraints

(e.g. overall budget cap or “no upfront payment”)



The Copilot then invokes a parsing step (ParseKeyInfoNode) that converts this into ParsedOrderRequirements — a normalized structure suitable for agent processing.

For example, free-form texts like “around 40 EUR” may be parsed into numeric budget ranges, or “no upfront” into a boolean flag, and the port-in date text into a standardized date format.

Once parsed, the Supervisor passes the ParsedOrderRequirements to the agent workflow. Each agent augments the data with its results:

The Portability Agent returns a PortabilityResult (eligible or not, any warnings)

The Offer Agent produces a list of planOptions (plans that match the criteria, with pricing)

The Device Agent returns deviceOptions (device models and installment details available)

The Insurance Agent returns insuranceOptions (coverage options and prices)

The Supervisor’s aggregation step combines these into a single OrderRecommendations object, which contains the curated recommendations to display. This recommendations payload is sent back to the front-end, where the Copilot UI renders it as interactive cards for the CSR.

After the CSR selects a plan/device/insurance combo, the choice is captured as SelectedOrderConfig (the IDs or codes of the chosen plan, device, etc.). The Copilot backend then maps this to the exact BSS order fields needed. In draft mode, it prepares either a BSSFillInstruction (for front-end field injection) or an OrderDraftDTO for the BSS’s order API.

Finally, the system populates the BSS order with these values (e.g. plan SKU, device SKU, number to port, insurance code) and returns control to the CSR for final submission.



show customer detail card of customer id 23210320135423.



Pandora

23210320135423

Active

Number Of Active Account: 3

Number Of Effective Agreement: 1

Current Bill: 1,201,88 kr.

unpaid off

Last 3 Months Consumption: 1201.88

Service Overview

B2B Normal Product Specification (1) Broadband (2)

Mobile & MBB (11) Multiplan (1)

Insurance ProductLine (7)

Contact Person (1)

1.Pandora AS

Recent complaints: 0

The agreement will expire in 0 days. [View Detail](#)

Ongoing orders: 7. [View Detail](#)

Recent new subscriptions: 22 [View Detail](#)

Recent terminated subscriptions: 0



Key Data Objects in the Copilot Workflow

Object	Stage in flow	System owner	Key fields (examples)	Contract notes
KeyOrderInfo	Intake	Order Genius Hub (Copilot UI)	sessionId: string, customerId: string, numberIntent: {portIn: bool, msisdn?: string, donor?: string, pac?: string}, planIntent: {budget?: number, usageText?: string}, deviceIntent: {model?: string, installment?: bool}, constraints?: {...}	Human-provided + assisted capture. May contain PII; apply masking rules in logs.
ParsedOrderRequirements	Normalize / slot parsing	Veris Agent Builder	msisdn?: string, portIn: bool, budgetRange?: {min,max}, deviceModel?: string, installmentPref?: enum, confidenceByField?: map	Normalized representation used by agents. Includes confidence/validation hints. Not written to BSS.
OrderRecommendations	Decide / recommend	Veris Agent Builder	planOptions[], deviceOptions[], insuranceOptions[], eligibilityFlags[], rationale?: string, rankScores?: {...}	Recommendation set only. Should include eligibility + rationale for audit. Not written to BSS.
SelectedOrderConfig	User selection	Order Genius Hub + Veris Agent Builder	selectedPlanId: string, selectedDeviceSku?: string, selectedInsuranceId?: string, portingDetails?: {...}, approvals?: {humanConfirmed: bool}	"Decision payload" chosen by user. Becomes input to draft write-back mapping.
OrderDraftWriteBack (draft payload)	Draft write-back	Veris Agent Builder - Veris Suite	orderDraftId?: string, fieldMappings: [{bssField, value}], warnings[], readyToSubmit: bool, validationResults?: {...}	This is the boundary object: what actually gets injected/written as a draft into Veris Suite. Final commit remains human-gated (unless explicitly configured otherwise)



Governance Checkpoints (Draft Mode & Human Approval)

Throughout this flow, **governance and control** are maintained to meet telecom compliance standards. Notably, the Copilot operates in a

“draft mode”

when interacting with the live BSS. All AI-driven actions culminate in a proposed order draft which the human CSR must approve and submit — the AI never directly finalizes the order without human sign-off.

In our example, after the CSR selects a recommended combination, the Copilot populates the order form but leaves it in an editable draft state. The CSR can verify every field and still has the opportunity to adjust anything before clicking the official **Submit** button.

This ensures that existing **BSS business rules and validation** are not bypassed: the final submission goes through the standard BSS order entry process (e.g. credit checks or policy validations happen as they normally would on submission). Certain steps in the agent flow can also be configured to require explicit confirmation.

For instance, if a high-value device was recommended but the customer’s credit is borderline, the system might flag it for CSR double-check or managerial approval before proceeding. All such **approval gates** are configurable.

In short

Agentic BSS augments the CSR’s workflow but does not remove the CSR’s authority — the human remains in control of the final decision, satisfying compliance and accountability requirements.

The **draft order** approach also means the core BSS data model isn’t altered; Copilot’s output is injected via official channels (e.g. using an Order Draft API or UI bridging) so that auditing and downstream processes see a normal order transaction.



Configurability via Playbooks

The entire scenario is driven by a **Playbook configuration** rather than hard-coded logic. Each supported business scenario (intent) in the Agentic BSS Copilot is defined in a **playbook** that declares its key parameters.

For the “**Mobile Port-In with Device**” use case, the playbook specifies the intent trigger (e.g. phrases indicating a new mobile connection), the required **slots** (information pieces) such as number port-in details, plan preferences, device need, etc., and which agents are involved in fulfillment. It also defines the sequence of agent/tool calls (for example: run PortabilityAgent and OfferAgent immediately after key info is parsed, then DeviceAgent if a device is needed, etc.) and how the UI should render the intermediate steps (like which **Copilot cards** to display for input and for recommendations).

Much of the dialog and prompting is configurable via playbook as well. If certain required slots are missing from the initial input, the system refers to a **slot-filling playbook** that provides domain-specific prompt templates to ask the CSR for that info in a concise, professional manner.

For example, if the desired port-in date was not provided, the playbook might supply a prompt like “To complete the port-in, could you confirm the preferred port-in date?”.

The behavior for fallback or alternate flows is also governed by playbook rules — e.g. if the customer said they don’t need a device, the DeviceAgent can be skipped entirely as defined by the scenario configuration.

Because playbooks are external configurations, **adding a new scenario or modifying an existing flow** (such as changing the order of checks, or what to do if a step fails) does not require recoding the agents. Playbooks support versioning and gradual rollout, so new AI-driven flows can be tested safely before full deployment.

This makes the system highly adaptable to different product offerings or policy changes: telecom business teams can update intents, slots, or tool integrations in the playbook to adjust the Copilot’s behavior without touching core code.



Error Handling and Fallbacks

Even with AI in the loop, the solution is built to handle errors and exceptions gracefully, ensuring a **telecom-grade** experience.

The architecture defines a clear error object model (codes and messages) for agent failures or API issues, along with fallback strategies at the conversation level.

If a required slot cannot be parsed from the CSR's input (say the budget phrase was unclear), the system doesn't throw an error — it records a null for that field and the Copilot politely asks the CSR a follow-up question to clarify (this is handled via the slot-filling playbook as noted).

If an external API call fails or times out (e.g. the device inventory service is down), the Supervisor agent will log the error and might **gracefully degrade** the recommendations:

For instance, proceeding with just plan and portability results, and showing a message in the Copilot UI that device suggestions are temporarily unavailable.

Each backend API call is given a reasonable time budget (often under 1.5 seconds); if exceeded, the call is aborted and a fallback is triggered so the CSR isn't left waiting indefinitely.

Common fallback behaviors include:

using cached data

(the system may cache recent product catalog and inventory info to use if live calls fail)

defaulting to a simpler flow

(e.g. if the recommendation step fails entirely, the Copilot can revert to the traditional manual input process for that portion)

These contingencies are part of the playbook definition and agent orchestration logic.

Importantly, all errors or overrides are surfaced to the CSR in a user-friendly way

— for example, if the number port-in check agent returns that the PAC code is invalid, the Copilot would prompt the CSR to re-confirm the code or offer to proceed without port-in.

The chat-style interface allows these corrections to happen naturally, guided by the AI.

Thanks to these safeguards, the system maintains robustness: a “happy path” may take 2–3 seconds to return results, but even in “unhappy paths” the CSR is supported with next steps rather than hitting dead-ends. 47



Logs, Observability and Metrics

To meet enterprise requirements, the **Agentic BSS Copilot** provides full **observability** into each session.

Every interaction is tagged with a unique **session ID** (as seen in the `sessionId` field of the `KeyOrderInfo`) that ties together the conversation, agent invocations, and resulting order.

The system maintains a persistent **Order Context** per session, which tracks the state and decisions made: the currently selected plan/device/insurance (if any), the latest parsed requirements, and even the history of recommendations shown to the CSR. This context, along with a detailed event log, is stored for audit and troubleshooting.

For instance, one can trace a single order flow across: the CSR's initial request, each slot-filling prompt and answer, the Portability/Offer/Device API calls made, and the final output applied to the order. Every agent's output is logged, and any errors include timestamps and error codes, ensuring telecom-grade auditability.

These logs feed into metrics that the business can monitor: **success vs. failure rates** of the Copilot-assisted flows, average cycle time, and the distribution of outcomes.

One key metric is slot fill rate (what percentage of required info was captured without additional prompts, versus how often the AI had to ask follow-ups).

Another is recommendation override rate — for example, how often the CSR accepts the top recommended plan vs. choosing an alternative or manually overriding.

A high override rate might indicate the recommendation model needs tuning. The system can also track if CSRs ultimately abandon the Copilot flow and revert to manual input (a critical measure of AI assist effectiveness).

Additionally, because each session and outcome is logged, continuous learning is possible: Cloudnet.ai's team can analyze transcripts and outcomes to improve agent prompts or playbook rules.

In production, these metrics and logs would be surfaced on a dashboard, giving operations a real-time view of Copilot usage, order conversion success, average handling time reduction, and any error trends.

By combining this telemetry with traditional BSS monitoring, the Agentic BSS implementation ensures that the AI-driven processes remain transparent, debuggable, and optimized over time — aligning with the “telco-grade, auditable, and compliant” principle of the platform.

07

Deployment and Integration Considerations

In This section

This section describes practical deployment and integration considerations for implementing Cloudnet.ai's Agentic BSS in a telecom environment.

It focuses on deployment environments, identity and tenant boundaries, data movement and logging, and the resilience patterns required for mission-critical operations, consistent with a control-plane / execution-plane separation.



Deployment environments

Cloudnet.ai's Agentic BSS is designed to be deployed in cloud, hybrid, or on-premises configurations to align with operator constraints on data sovereignty, latency, and regulatory posture.

This flexibility is consistent with modern telecom architecture guidance that emphasises cloud-native, modular deployment and provides explicit “deployment and runtime” guidance for turning reference architectures into real-world solutions.

A practical deployment framing is:

Cloud deployment (control plane in cloud):

Veris Agent Builder (agent orchestration/runtime) and Order Genius Hub (copilot control surface) can run in cloud environments where operators want rapid iteration

This approach relies on exposing the execution plane (systems of record) through secure APIs and event streams.

Hybrid deployment (split control and execution adjacency):

Control plane services can run in cloud while tool connectors, event gateways, or sensitive processing runs closer to systems of record (often within an operator-controlled environment).

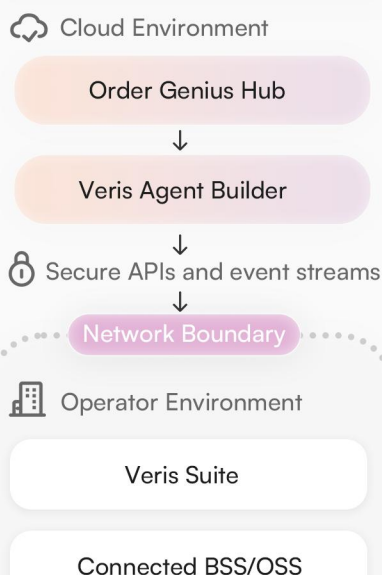
This is commonly used when operators want cloud agility but need tighter control over data movement or low-latency links to OSS/BSS.

On-premises deployment (operator data centre):

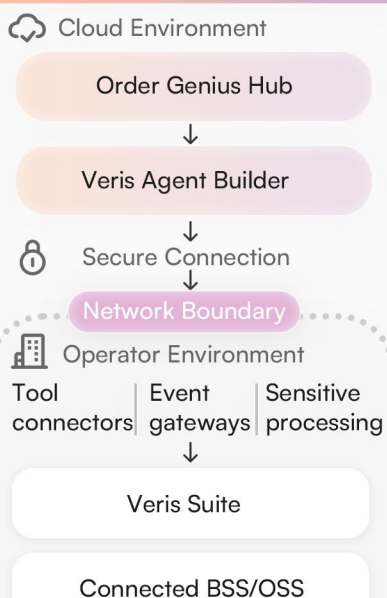
For strict sovereignty or latency requirements, the orchestration layer and integration gateways can run within operator infrastructure.

Modern deployment guidance for telecom component architectures explicitly assumes a containerised runtime: ODA deployment and runtime guidance highlights a Kubernetes-based execution environment and reference implementation patterns that can run on “bare metal, virtual machine, Cloud IaaS or Cloud PaaS,” provided they support a Kubernetes cluster.

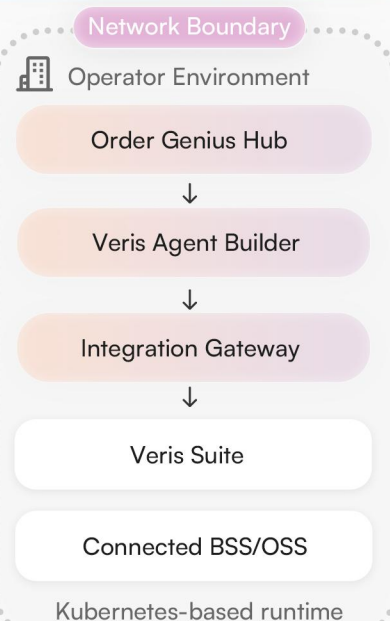
Cloud deployment



Hybrid deployment



On-premises deployment





Identity, access control, and tenant boundaries

Agentic execution introduces a new class of “actors” (agents and tool-service principals) that **must be governed as rigorously as human users.**

The security posture should assume a modern, componentised attack surface and avoid relying on perimeter-only controls; ODA security governance explicitly notes that component-based approaches can increase attack surfaces and recommends DevOps-driven controls and **zero-trust principles.**

In practice, identity and access control should be implemented with **three clear separations:**

● **Separate human identities from agent identities:**

CSRs and supervisors authenticate via the operator’s identity provider; agents run under service principals with explicit scopes.

This reduces lateral movement risk and makes approval/attribution unambiguous.

● **Least privilege and privileged-function logging:**

Access must be scoped per “tool” and per action (read vs. write), with explicit controls on elevated operations and write-back.

This aligns with National Institute of Standards and Technology SP 800-53 controls that emphasise least-privilege enforcement and audit/logging expectations for privileged or security-relevant actions, including logging the execution of privileged functions.

● **Strong authentication, continuous evaluation, and audit trails:**

A zero-trust posture treats every access request as requiring explicit authentication and authorisation, and benefits from logging each request to support investigation and compliance evidence.

Guidance from the Canadian Centre for Cyber Security emphasises evaluating and logging each access request to support continuous compliance and auditing in a zero-trust approach.

From an integration standpoint, operators typically federate identity using open standards (e.g., SAML or OpenID Connect) and delegate authorisation using OAuth 2.0-based flows; these standards provide well-defined semantics for authentication assertions and limited-scope access to HTTP services.

Tenant boundaries should be explicit and enforceable across: tool endpoints, caches, logs/traces, and evaluation datasets. Where multiple brands or business units exist, treat each as a tenant boundary for both data and operational controls.



Data movement, caching, and logging

Agentic BSS only scales when data movement is governed and minimised.

A practical rule set is:

System-of-record integrity:

Authoritative customer, product, order, and billing objects remain in the system of record (execution plane).

Control plane components should not become shadow systems of record.

Data minimisation and controlled caching:

Cache only what is required for latency and journey continuity (e.g., short-lived session context, non-sensitive reference data, idempotency tokens).

Keep caches time-bounded and tenant-scoped.

Logging that is audit-grade but privacy-aware:

Agents should log decision artefacts, tool call metadata, and approval events so incidents can be reconstructed end-to-end.

At the same time, audit trails can create privacy risk if they contain personally identifiable information; notes the potential to reveal PII in audit trails depending on what is recorded.

For AI-enabled systems specifically, governance guidance stresses retention and provenance documentation as part of a defensible audit posture.

National Institute of Standards and Technology's Generative AI Profile recommends updating or establishing record-retention policies to preserve sufficient history for auditing and investigation, and setting transparency/provenance practices to document the origins of training data and generated outputs—both directly relevant to agent observability, trace completeness, and post-incident reconstruction.

Practically, this implies two logging tiers:

Operational telemetry:

latency, error rates, tool invocation counts, and outcomes (PII-minimised).

Audit log:

approvals, policy decisions, write-back payload hashes/IDs, and traceable tool-call context (redacted where appropriate, with controlled access).



Latency, reliability, and operational resilience

Telecom-grade workflows require predictable behaviour under partial failure. A governed agent layer must therefore implement failure containment and safe degradation, not “best-effort” retries.

A resilient posture includes:

Timeouts and bounded retries per tool:

Treat each tool/API call as a managed dependency.

Timeouts prevent indefinite waits; bounded retries avoid cascading load during upstream incidents.

Circuit breaking and back-pressure:

When a downstream dependency exhibits elevated error rates or tail latency, the orchestration layer should reduce calls, shift to draft-only mode, queue work, or escalate to human review until the dependency recovers.

Idempotency for write operations:

Execution-plane writes should be protected with idempotency keys so retried commits do not create duplicate orders, duplicate billing actions, or inconsistent states.

Observability built-in:

Production deployments should provide trace-level observability across the service mesh and tool adapters.

TM Forum’s ODA Reference Implementation explicitly calls out observability in the runtime environment and positions identity management and a service mesh as baseline operational capabilities for ODA deployments.

Controlled fallbacks:

When automation cannot proceed safely (missing data, uncertain policy classification, dependency outage), the system should degrade to draft-only execution, route to human approval, or produce a structured “what failed / what is missing / what to do next” packet to minimise rework.

These mechanisms collectively ensure that Agentic BSS automation remains dependable under real operational conditions—high concurrency, heterogeneous dependencies, and routine exceptions—while preserving governance, auditability, and system-of-record integrity.



Risks, Limits, and Mitigations

08

In This section

Deploying an Agentic BSS in production is not merely selecting a better AI model—it is an **operating-model transformation** spanning data, integration, governance, and people.

Analysts rightly question whether "agentic" claims hold up under real telco constraints, with some explicitly cautioning that



over

40%

of agentic AI projects may be cancelled by end-2027 due to rising costs, unclear business value, and insufficient risk controls.

This highlights the need for **disciplined guardrails** rather than unchecked experimentation.

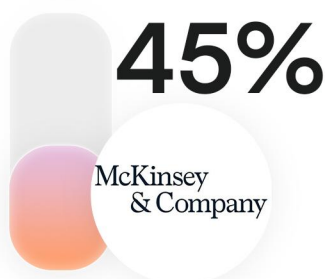
This section lays out the primary risk areas and the concrete mitigations that Cloudnet.ai's Agentic BSS employs, aligned to authoritative guidance from on modular, API-led modernisation and security governance, and from on AI risk management and auditability.



Data readiness and quality

Risk:

AI-driven orchestration is only as good as the data it uses. Telecom BSS data is often inconsistent across domains (customer, product catalogue, inventory, billing), and agents may infer incorrectly if required fields are missing or rules are ambiguous.



McKinsey notes that **data limitations are a top inhibitor** to scaling AI agents; in one survey, more than 45% of telecom executives cited poor or incomplete data as a barrier.

In practice, data gaps can cause agents to produce invalid orders, mis-route tasks, or simply stall, leading to increased order fallout and manual interventions.

Mitigations (practical controls):

Cloudnet.ai recommends treating data reliability as a first-class engineering and operating concern:

Playbook-enforced data contracts:

Define required fields/format in each journey step. The agent verifies schema compliance during the validate stage, preventing bad commits.

Real-time validation and feedback:

On detecting missing or inconsistent data, the system prompts for clarification before proceeding. For example, if a CLI eligibility check fails, the agent can halt and request the missing value, rather than blindly continuing.

Canonical source-of-truth retention:

Authoritative state (e.g. CRM customer record, product catalog) remains in the execution plane (Veris Suite). The control plane only caches short-lived context, reducing drift risk.

Data-quality telemetry and remediation:

Track metrics like field completion rates and fallout causes. These feed back into data-governance processes so that systemic data issues are fixed upstream.



Integration complexity and exception handling

Risk:

Telecom order fulfilment spans many systems (CRM, CPQ, provisioning, billing, etc.). Each integration adds potential failure modes. A single broken API or race condition can fragment a transaction across multiple systems.

For instance, an inventory reserve error might leave an order partially committed, requiring a manual rollback.

TM Forum's Open Digital Architecture aims to simplify such integration, but componentization also increases integration points.

Industry literature (e.g. Oracle's OSS documentation) characterises these "fallout" orders as a common operational bottleneck. Without robust handling, automation can actually raise the volume of escalations.

Mitigations:

Agentic BSS is designed for reliable integration:

Typed tool contracts and adapter layer:

Every backend function (eligibility check, order submit, network reservation) is invoked through a well-defined "tool" with strict input/output schemas. This minimizes ambiguous or unhandled errors.

Event-driven retry and dead-letter queues:

If a tool call fails (e.g. provisioning timeout), the event bus triggers a remediation flow.

Transient errors are retried automatically; non-recoverable issues generate alerts or escalation tickets.

Idempotency and transaction keys:

Write operations include idempotency tokens to prevent duplicates in case of retries.

The control plane tracks each operation's state, enabling safe replay or rollback.

Progressive autonomy rollout:

Start integrations in read-only or draft mode. Only when confidence is proven (through monitoring of call success rates and fallback counts) are critical writes enabled for automation.



Change management and human adoption

Risk:

Even a technically perfect system fails if users don't trust it. Telecom operations are complex and historically manual; frontline staff may resist black-box automation.

McKinsey
& Company

McKinsey observes that productivity gains from early AI pilots often erode when rolled out broadly, due to inadequate training, lack of executive buy-in, or misalignment between tech and process.

Many pilot projects stall simply because the human users circumvent or ignore the new tools.

Mitigations:

Cloudnet.ai recommends an adoption model that treats governance and usability as part of the product:

Draft-first, co-pilot experience:

The Order Genius Hub presents AI suggestions as drafts that the CSR can review and modify.

This builds familiarity and trust (and guarantees a human always sees the result).

Explainable hints and rationale:

Where possible, the agent provides brief justifications ("Recommending plan upgrade because XYZ eligibility is met"). Explaining why increases user confidence.

Stakeholder involvement and training:

Cross-functional workshops define the new workflows. Early adopters and "super users" are enlisted as champions.

Training materials highlight how the copilot saves time and reduces errors.

Slow autonomy ramp-up:

Use the metrics from Section 9 (e.g. rollback rate, user overrides) to decide when to escalate autonomy.

Start with simple tasks (form filling) before moving to decision-heavy steps.



Model risk: hallucination, confabulation, and tool misuse

Risk:

Generative models can “hallucinate” plausible-seeming but incorrect outputs, a known concern for decision systems.

NIST’s Generative AI Profile explicitly warns of **confabulation** and recommends robust controls:

continuous monitoring, and verifying impacts when AI outputs feed into downstream actions.

A hallucinated field value (like a fabricated discount) could corrupt an order.

Furthermore, without guardrails an agent might misuse a tool (e.g. calling a cancellation API instead of change). Gartner warns that without careful validation, agentic AI can stray from intended goals, eroding value.

Mitigations:

Layered defenses around model outputs:

Strong validation before actions:

All agent proposals undergo deterministic policy and rules checks. For example, a recommended billing adjustment is cross-checked against actual invoices and tariff rules before execution.

Restricted tool scopes and allowlists:

Agents have no carte-blanc. Each tool interface explicitly bounds what the agent can request. High-risk operations (account write-off, service cancel) may require supervisor approval even if proposed by the agent.

Retrieval-augmented grounding:

When agents generate text, they cite authoritative sources (e.g. CRM data, SLA clauses). This anchors outputs to facts and highlights mismatches early.

Red-teaming and continuous evaluation:

Model behavior is stress-tested in sandbox. NIST recommends test-evaluate-verify loops; Cloudnet.ai maintains a validation track for model outputs to catch drifts or vulnerabilities post-deployment.

Fallback to human oversight:

If the model’s confidence is low or unusual patterns are detected, control reverts to a human operator. No action is committed unless safety checks pass.



Compliance, security, and auditability

Risk:

Telecom systems are highly regulated (privacy, lawful intercept, SLA reporting). Introducing AI agents adds new audit, security, and privacy dimensions.

TM Forum notes that shifting to component-based architectures can **increase attack surfaces**, requiring zero-trust governance.

Organizations must also comply with PIPEDA (Canada) and other data laws.

NIST SP 800-53 mandates logging of privileged actions, and its Generative AI profile calls for long-term retention of decision logs.

Without proper controls, automated actions could bypass existing audit trails or expose sensitive data.

Mitigations:

Built-in compliance posture:

End-to-end audit trail:

Every agent action is logged—inputs, chosen actions, tool calls, results. Immutable logs record which identity (human or agent) approved each commit.

Role-based access and segmentation:

Human users and agent services have separate identities. Least-privilege access policies ensure agents cannot query customer PII unless needed, and cannot modify records outside scope. Tenant data is strictly partitioned.

Privacy and data handling:

Only non-sensitive data is stored in logs; PII in context is redacted or hashed. Cloudnet.ai follows NIST guidance on data provenance, maintaining logs that tie back to source records without leaking private fields.

Policy enforcement:

Enterprise policies (change windows, spending limits, SLA credits) are codified in the control plane. Any action that would violate a policy is automatically rejected or escalated for human review.



Cost and latency control

Risk:

Large-scale AI incurs non-trivial cost (compute for model inference, API usage, data pipelines) and can add latency to user-facing operations.



Gartner

Gartner's warning about project cancellations cited “**inadequate financial planning**” as a failure driver.

In practice, unbounded AI queries or chat interactions can blow past budget, and slow model responses (even a few seconds) can frustrate users. For telcos striving to lower cost-to-serve (PwC notes slim margins and growth under 3%), undisciplined AI can undo those gains.

Mitigations:

Explicit monitoring and throttling:

Per-journey budgets:

Each agent journey is assigned a maximum cost and latency budget. The system tracks token counts and API calls, halting or simplifying responses when approaching limits.

Efficient model routing:

Lower-cost models handle simple extraction or classification tasks. The most powerful (and expensive) models are invoked only when needed for complex decision-making.

Caching and asynchronous work:

Common queries (e.g. tariff lookups) are cached. Lengthy operations (bulk order checks, bill runs) are conducted asynchronously or via batch events to avoid blocking the user session.

Live monitoring and alerts:

Telemetry tracks per-transaction cost, queue lengths, and 95th percentile latency. Exceeding thresholds triggers immediate alarms or temporary throttling of AI features until capacity is scaled.



Across all these areas,

Cloudnet.ai's approach emphasizes **measurement and feedback**. Each automated action is tagged so operators can measure its impact (Section 9), and mitigations are iteratively improved based on real-world data. This KPI-driven, risk-aware rollout is critical to proving value and building analyst confidence in Agentic BSS.



Evaluation and Measurement Framework

09

In This section

To ensure Cloudnet.ai's Agentic BSS is **trustworthy, auditable, and telecom-grade**, we implement a comprehensive evaluation and measurement framework.

This framework spans offline testing of AI agents, **online KPI monitoring** in production, strict **safety governance metrics**, robust **observability** for debugging/audit, and proactive **cost controls**.

The multidimensional approach goes beyond simple success/failure checks — it measures accuracy, efficiency, autonomy, and compliance, aligning with industry best practices for telco-grade performance.

By continuously tracking these metrics, Cloudnet.ai can demonstrate that Agentic BSS operates with carrier-grade rigor while continuously improving through data-driven tuning.



Offline Evaluation Metrics

Before deployment, Agentic BSS agents are rigorously evaluated on curated test scenarios to validate their core capabilities.

Key offline metrics include:

1

Parsing Accuracy

(Slot Extraction & Structured Intent):

Measuring how precisely the AI parses free-text KeyOrderInfo into structured ParsedOrderRequirements. Each relevant slot (e.g. port-in number, budget range, device choice) is compared against a ground truth for correctness.

We use standard NLU evaluation methods (precision/recall or F1) to ensure the agent's natural language understanding meets high accuracy.

For example, if a test order says "I need to port my 555-1234 number to a \$50 5G plan," the agent must correctly extract all required details into the order schema.

High parsing accuracy gives confidence that downstream automation will have the right data.

Any mis-extractions trigger refinements to the prompt logic or slot model before go-live, aiming for telecom-grade precision in order capture.

2

Recommendation Quality

(Accuracy & Override Rate):

Evaluating the quality of the agent's suggestions or decisions (e.g. recommended plans or actions) against either known optimal outcomes or expert judgment. Accuracy here means how often the agent's recommendation matches the correct or ultimately chosen solution.

In offline replay tests, we compare the agent's recommended offer or resolution to what an experienced human **would** select. We also track the **override rate** — how frequently a human (or customer) ended up overriding the AI's recommendation. A low override rate (equivalently, a high decision adoption rate) indicates the agent's choices are usually accepted, reflecting strong decision-making alignment with business rules.

If the override rate in tests is above a threshold, it flags scenarios where the agent's logic needs adjustment (e.g. updating the recommendation model or adding business constraints) to improve alignment before deployment.

Tool-Call Correctness

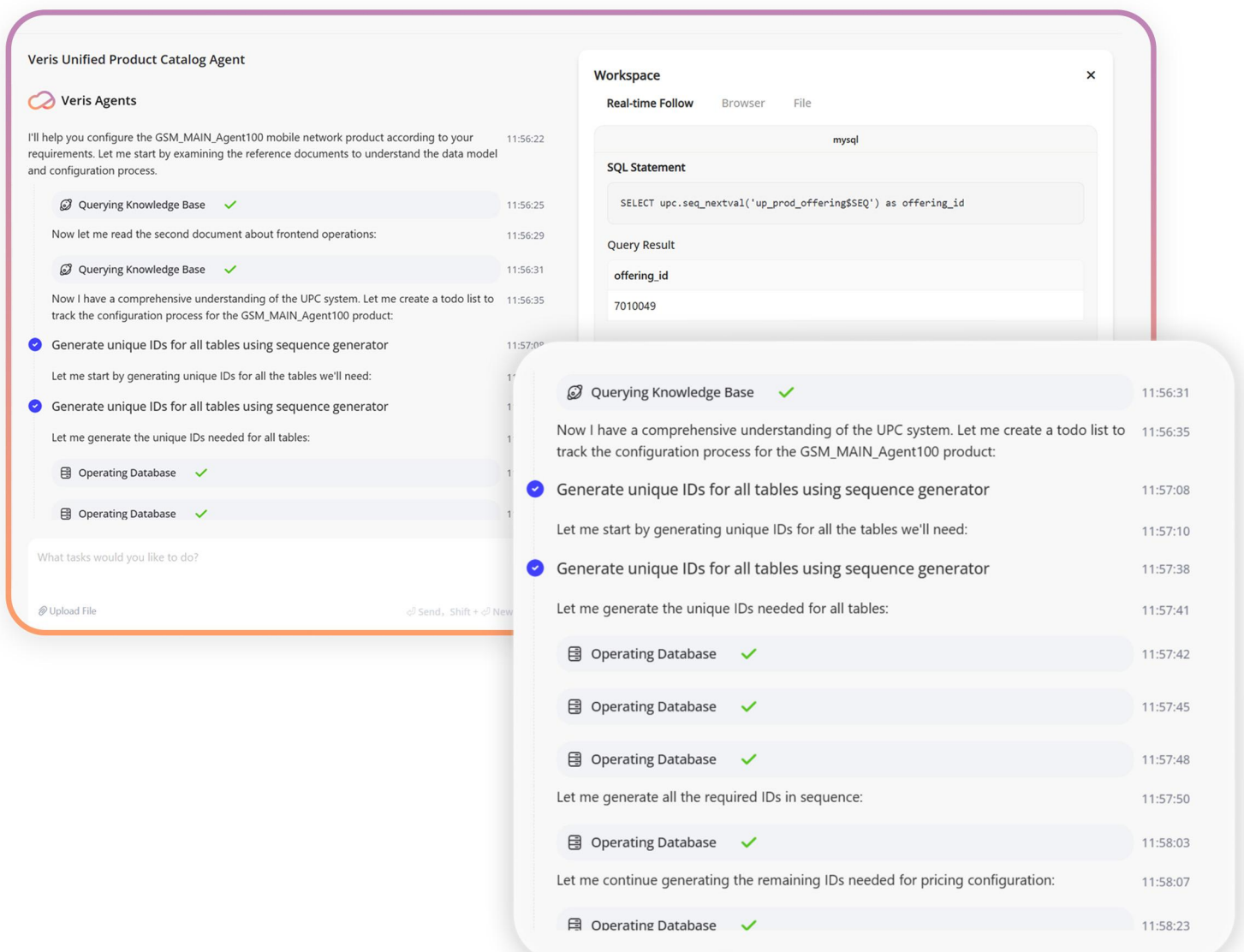
(Parameter Validity & Error-Free Rate):

Verifying that every external API or tool invocation by the agent is correct and produces the expected outcome. In a lab setting, the agent's workflow (which might call BSS APIs, product catalog services, etc.) is executed on sample orders.

We inspect each tool call for valid parameters, correct use of APIs, and proper handling of responses. Metrics include the percentage of calls with **valid parameters** (no format errors or missing fields) and the **error-free execution rate** of these calls. Essentially, this gauges the agent's "action selection accuracy" — did it choose the right action and use it correctly at each step? A high score means the agent's integrations are correctly configured and robust.

Any tool-call errors (e.g. calling the wrong API or sending malformed data) are caught in offline testing and corrected.

The goal is to reach near 100% correct tool usage in tests, ensuring smooth end-to-end execution when the agent goes live.



The screenshot displays the Veris Unified Product Catalog Agent interface. The main chat window shows a conversation with the agent, including a list of tasks and a workspace window.

Veris Unified Product Catalog Agent

Veris Agents

I'll help you configure the GSM_MAIN_Agent100 mobile network product according to your requirements. Let me start by examining the reference documents to understand the data model and configuration process. 11:56:22

Now let me read the second document about frontend operations: 11:56:29

Now I have a comprehensive understanding of the UPC system. Let me create a todo list to track the configuration process for the GSM_MAIN_Agent100 product: 11:56:35

- Generate unique IDs for all tables using sequence generator 11:57:08
 - Let me start by generating unique IDs for all the tables we'll need: 11:57:10
 - Let me generate the unique IDs needed for all tables: 11:57:41
 - Operating Database ✓ 11:57:42
 - Operating Database ✓ 11:57:45
 - Operating Database ✓ 11:57:48
 - Let me generate all the required IDs in sequence: 11:57:50
 - Operating Database ✓ 11:58:03
 - Let me continue generating the remaining IDs needed for pricing configuration: 11:58:07
 - Operating Database ✓ 11:58:23

What tasks would you like to do?

Upload File Send, Shift + New

Workspace

Real-time Follow Browser File

mysql

SQL Statement

```
SELECT upc.seq_nextval('up_prod_offering$SEQ') as offering_id
```

Query Result

offering_id
7010049



Online KPIs for Operational Impact

Once deployed, Cloudnet.ai continuously monitors live performance through key KPIs, reflecting real-world operational impact. These online metrics demonstrate how Agentic BSS improves efficiency and customer experience, in line with telecom industry benchmarks (e.g. TM Forum definitions for service KPIs).

Major online KPIs include:

● Average Handling Time (AHT):

The average duration to complete an order or resolve a request when handled by the AI agent (including any conversational steps and back-end processing).

This is analogous to call center AHT — a shorter handling time means greater efficiency. We track AHT for agent-managed sessions and compare it to baseline manual handling times.

For example, if a mobile plan change traditionally takes 10 minutes with a human CSR, but the Agentic BSS can complete it in 6 minutes, that reflects a significant efficiency gain.

AHT is measured from the start of the interaction to final completion, including any agent-to-agent or agent-to-human handoffs. Reducing AHT (while maintaining quality) is crucial for operational efficiency.

Our target is to meet or exceed industry norms (e.g. telecom contact centers often aim for ~5—7 minutes AHT) by streamlining workflows through AI automation.

● Order Fallout Rate:

The percentage of orders that fail to complete automatically and require manual intervention or rework. In telecom, order fallout refers to orders that encounter errors during processing — for example, due to missing data, system timeouts, or rule conflicts.

We calculate fallout rate as failed orders / total orders, on a rolling basis. A low fallout rate indicates robust end-to-end automation. Agentic BSS is designed to reduce fallout by handling exceptions or requesting clarification when data is missing, rather than simply failing.

We instrument the system to log any order that cannot be completed by the agent so it can be analyzed and fixed.

For instance, if the agent lacks a rule for a corner-case plan combination and an order falls out, that case is flagged for improving the agent's knowledge.

Cloudnet.ai targets a fallout rate as low as possible (e.g. <1—2%), comparable to best-in-class order management systems. This metric directly ties to revenue and customer satisfaction, as high fallout means delays and manual fixes.



First Call Resolution (FCR):

The percentage of customer requests or issues resolved on the **first contact** (or first session) without requiring the customer to re-contact support.

This is a critical customer experience metric — a higher FCR means customers get their issue resolved in one go. For Agentic BSS, FCR measures how often the AI agent fully resolves the inquiry (for example, successfully completing an order or troubleshooting a problem) within the initial interaction, with no follow-up calls or escalations needed. By automating end-to-end tasks, the agent aims to **improve FCR**.

We define and measure FCR in line with standard practice: if a customer does not contact again about the same issue within a defined period (e.g. 7 days), it counts as resolved on first contact.

Our live dashboards show FCR trending over time; improvements in FCR after deploying Agentic BSS indicate the AI is effectively handling requests that previously might have required multiple touchpoints.

Containment Rate (End-to-End Automation):

This KPI tracks the proportion of orders or support cases that are **completed with no human intervention at all**.

In other words, the AI agent contained the entire interaction and fulfillment within the automated system. It is similar to FCR but focused on **internal automation** rather than customer perception.

A contained order means the agent never had to escalate to a human nor needed any manual step. We measure containment rate = (orders completed by AI alone) / (total orders initiated with AI). A high containment rate signifies true autonomous operations.

For example, if 90 out of 100 incoming mobile activations were handled entirely by the agent (from capturing requirements to provisioning and confirming), the containment rate is 90%.

This metric is crucial for quantifying labor savings. We aim to push containment as high as possible while staying within policy bounds (some cases will intentionally require human approval).

Even when a human is in the loop, the agent tries to maximize what it automates.

Completion rate without manual intervention is a recognized KPI for AI agents, and our system continuously reports it.

Any drop in containment triggers analysis into whether new scenarios are arising that the agent wasn't trained for (prompting us to expand its capabilities).



● Latency (Response & Completion Time):

We monitor the system's speed in two dimensions — **response latency** and **end-to-end completion time**. **Response latency** is how quickly the agent responds to user inputs or events (e.g. the time to present the next dialog message or confirm an action after the user provides information).

Completion time is the total elapsed time to finish the entire order or task. Low latency is a must for good UX and meets real-time operational needs. We instrument each agent action with timestamps to track processing delays.

This includes the latency of tool API calls, the LLM or reasoning steps, and any waits (e.g. for an external system).

The framework sets thresholds (for example, a response to the user should typically be under 2 seconds, and a full order completion perhaps under 1 minute depending on complexity)

If latency exceeds targets, alerts are raised to investigate bottlenecks. By analyzing latency metrics, Cloudnet.ai ensures the Agentic BSS responds and completes tasks within **telecom-grade** time limits (telecom operations often have strict SLAs for responsiveness).

Consistently fast performance is critical for AI agents to be viable in live customer-facing scenarios.

Together, these online KPIs give a real-time view of how Agentic BSS impacts operations — from efficiency gains (lower AHT) to success rates (higher FCR/containment, lower fallout) and speed (latency).

They are visible via operational dashboards and are aligned with known industry KPIs for customer experience and process efficiency. Cloudnet.ai's framework allows benchmarking these metrics against **baseline values** and **target thresholds**, ensuring the AI meets or exceeds traditional performance standards.

For example, improvements in FCR or reductions in AHT directly translate to better Net Promoter Scores and cost savings, which we track to demonstrate ROI.



Safety and Governance Metrics

In a telecom environment, trust and compliance are paramount. Cloudnet.ai therefore monitors a set of safety and governance metrics to ensure the autonomous agents behave within approved bounds and that there is always a fail-safe if something goes off course.

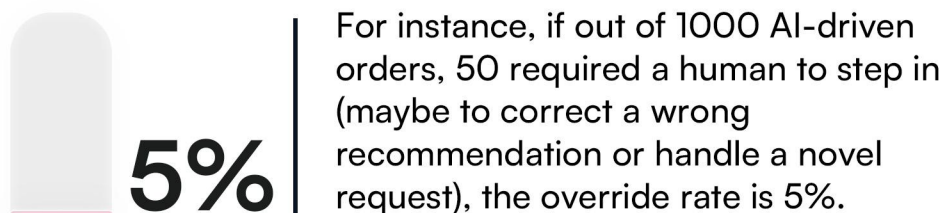
These metrics keep the system auditable and controllable, preventing the AI from straying beyond its mandate:

● Human Override Rate:

Measures how often a human supervisor or agent (e.g. a Customer Service Representative overseeing the AI) needs to intervene or override the AI's decision.

Every time the AI's suggested action is stopped or changed by a human (perhaps via a "manual approval" step or an emergency halt), it's logged. A low override rate indicates the AI is operating confidently and correctly, whereas a spike in overrides could signal mistrust or systematic errors.

Cloudnet.ai's platform **captures the human override rate at each fallback point.**



We aim to minimize this through continuous training, but we want humans to jump in whenever the agent is uncertain or likely wrong — thus override rate is an important safety valve indicator.

A sudden increase would trigger investigation into the agent's recent decisions (and likely a retraining or rule update to fix the underlying issue).



Rollback Rate:

The frequency at which the system had to undo or roll back an action initiated by the AI agent.

In telecom operations, if an automated process makes a change (provisioning a service, updating a record) that later turns out to be incorrect or harmful, a rollback procedure is triggered to restore the previous state.

We treat any AI-initiated rollback as a critical event to track.

Rollback rate is calculated as number of rollbacks / total automated actions.

For example, if the agent auto-processed 500 changes this week and 5 had to be reverted, rollback rate = 1%.

A low rate signifies the agent “gets it right the first time” in the vast majority of cases. Each rollback is analyzed in post-mortems to understand if it was due to a flaw in agent logic, a downstream system error, or missing validation.

Our goal is to keep this metric near-zero by building robust pre-checks (e.g., simulate the action in a dry-run or double-check policies before execution).

This metric assures that the Agentic BSS’s autonomous actions are **reliable and safe**, as frequent rollbacks would indicate instability.

Policy Violation Rate:

Tracks how often the agent attempts an action that breaches predefined policies, compliance rules, or business constraints. Agentic BSS operates under strict guardrails (for example, it cannot give unauthorized discounts, must follow data privacy rules, and respect credit check requirements).

A **policy violation** could be an attempted tool call that the agent is not permitted to make, an offer that doesn’t align with governance rules, or any execution that triggers a compliance flag.

We log every such incident as a violation (even if the platform’s guardrail blocked the action in time). The policy violation rate is ideally **zero**, since the agent is built to know its limits. In practice, especially during early stages or new skill rollouts, we may see a few violations which indicate where the agent’s knowledge or constraints need refinement.

For instance, if the agent tries to apply an expired promotion or access a customer segment it shouldn’t, those attempts are caught and recorded.

Monitoring this metric ensures **actions follow enterprise rules and regulations**. Cloudnet.ai’s governance engine will immediately halt any violative action and alert compliance officers.

Regular audits of violation logs help update the agent’s policies. This metric provides assurance to telco compliance teams that the AI is operating within the same rules a human employee would.



● Out-of-Scope Execution Attempts:

Counts the instances where the agent tries to perform tasks outside its authorized scope or domain. Each agent in the BSS is provisioned with certain capabilities and a scope of authority.

If it attempts something out-of-scope — for example, calling a database it shouldn't, modifying an unrelated system, or handling an inquiry type it wasn't designed for — the system flags it. These could be due to unexpected user requests or errors in the agent's reasoning. All such attempts are blocked and logged.

We monitor this to understand how often the AI is encountering inputs it cannot handle or is misunderstanding instructions. A high out-of-scope attempt rate might indicate the need to expand the agent's training data or to improve its intent recognition.

Crucially, this metric ties into **misbehavior detection**: by embedding health checks and validation at runtime, the platform can detect when an agent veers off its expected behavior and sandbox it.

For example, if an Order Agent unexpectedly tries to access HR data (clearly out of scope), that attempt would be stopped and recorded.

By reviewing these logs, we continuously refine the agent's boundaries and understanding, thus preventing rogue actions. This is another metric we strive to keep near-zero, demonstrating that the agent stays reliably on-task.

These safety and governance metrics ensure **human oversight and control** remain in place. The system is designed such that any time one of these metrics moves in an unfavorable direction (e.g., override or violation rates rising), Cloudnet.ai can proactively pause or dial back the agent's autonomy and investigate.

In practice, a governance board or AI ethics committee at the telco can regularly review these metrics via reports. The presence of override, rollback, and violation monitoring is a key reason Agentic BSS is **auditable** — every questionable action leaves a trace and can be accounted for.

This gives operators and regulators confidence that even as processes become autonomous, they remain **accountable and compliant**.



Observability and Debuggability

To maintain a telco-grade platform, Cloudnet.ai emphasizes rich observability and easy debuggability of the Agentic BSS.

Every action and decision made by the agent is traceable, enabling engineers and auditors to reconstruct what happened in any given session.

Key features of our observability framework include:

● Session and Trace IDs:

Every customer session or order execution is assigned a unique Session ID, and every step the agent takes carries a Trace ID (or a correlated ID) that links it back to the overall session. These IDs propagate across all microservices and API calls involved.

This means all logs, from the NLU parse to final fulfillment, can be aggregated by session. If an issue arises (e.g., an order was processed incorrectly on a certain date), we can instantly pull the full trace of that session using its ID.

This end-to-end correlation is crucial for **continuous monitoring** in complex BSS environments. It gives us a

”
black box recorder

for the AI's operation — a foundation for both real-time alerts and post-mortem analysis.



Agent Decision Logs:

The platform maintains detailed **agent logs** that capture each decision and reasoning step the AI agent makes. This includes the inputs the agent considered (e.g. the ParsedOrderRequirements, current customer context), the outcome of its internal logic or LLM reasoning (such as the content of a prompt or intermediate thought process), and the action decided upon at each node of the workflow.

Essentially, we log every agent “thought” and decision with a timestamp and explanation.

For instance, an agent log entry might record: “Step 3: OfferAgentNode — received budget= \$50, needDevice=true. Decided to call Offer API with planBudget=50 (based on rule X).”

These decision logs enable **audit replay** — one can review or even replay the session to see why the agent did what it did. The logs are structured and indexed by session/trace ID for quick search.

By capturing the agent’s reasoning, we treat the AI’s decision process with the same scrutiny as any business rule engine. This transparency is key to building trust. If a decision was wrong, developers or analysts can pinpoint where the reasoning went astray (e.g. a knowledge gap or a logic branch) and fix it in the next iteration.

API Call Logs

(Inputs, Outputs, Timestamps):

Every external call made by the agent to tools or APIs (for example, a CRM query, inventory check, or billing adjustment) is logged at a fine-grained level. The log record includes the exact **request parameters** the agent sent, the **response** received (or error code if it failed), and timing information (request start/end timestamps).

This is invaluable for debugging integration issues and ensuring data integrity. If an agent reports “SIM activation failed,” the support engineer can look at the API call log and see, for example, that the provisioning API returned a specific error due to an invalid SIM ID. All API calls are tagged with the session ID as well, so they slot into the overall trace.

We also log the **duration** of each call, which feeds into latency metrics. By analyzing these logs, we can identify slow services or error-prone APIs and work with those system owners to improve them.

Importantly, these detailed logs make the system **auditable** by external parties too — an auditor could verify that “on Sept 1 at 10:05, the agent queried database X with customer Y’s data and got response Z,” which supports compliance requirements (for example, proving that no unauthorized data access occurred).

This level of logging reflects the telecom operations mindset of thorough record-keeping and is aligned with frameworks like OpenTelemetry for distributed tracing.



Dashboard Visibility and Audit Replay:

Cloudnet.ai provides real-time dashboard tools for operations teams to visualize these metrics and traces. Key metrics (AHT, FCR, containment, etc.) are displayed in an executive dashboard (using industry-standard tools like Grafana or Power BI for live KPI tracking). Additionally, there's a console to inspect sessions in detail

— for example, a support engineer can search for a specific Order ID or customer interaction and retrieve the full timeline of that session: from the initial intent detection and KeyOrderInfo.parse, through each agent step, to final outcome.

The dashboard can present a **trace visualization** (e.g, a flow diagram with timestamps for each node execution — placeholder for such a visualization is noted above). This allows teams to **replay** the agent's execution path after the fact, either by reading the logs or even running a simulation through an “audit mode.”

Such replayability means if a dispute or anomaly arises (say a customer claims their plan change wasn't done correctly), the telco can play back the agent's actions to see exactly what transpired. Moreover, having all this in a user-friendly dashboard ensures that even non-developers (like operations managers or compliance officers) have **full visibility** into the AI's behavior and performance. They can spot trends (like rising override rates or slowdowns) and drill down on demand.

This observability framework, leveraging trace IDs and extensive logging, embodies the

”
telemetry and audit trails embedded

approach recommended for telco-grade AI systems . It not only aids debugging but also builds confidence with stakeholders that nothing in the “AI black box” is truly hidden — every decision is tracked and can be explained.

In summary, our observability and debug capabilities turn the Agentic BSS into a **glass box** rather than a black box. The combination of traceable logs, unique IDs, and dashboard tools means the autonomous agents are **fully accountable**.

Should anything go wrong, we can rapidly root-cause it. This level of transparency is essential for telecom operations, where regulatory compliance (e.g., audits, data lineage) and uptime requirements demand that every element of the system can be monitored and trusted.

By designing observability in from day one, Cloudnet.ai ensures Agentic BSS can be adopted without sacrificing the strict oversight and diagnostic practices telcos expect from any critical system.



Cost Monitoring and Controls

Autonomous AI agents introduce new cost vectors (such as AI compute usage) into BSS operations. Cloudnet.ai's framework includes careful cost monitoring and controls to keep operations **efficient and sustainable at scale**.

We track how resources are consumed and put caps or optimizations in place, so that the Agentic BSS delivers ROI without budget surprises.

Key cost-related metrics and controls include:

● Token Usage (LLM Consumption): ● API Call Frequency and Utilization:

If large language models (LLMs) are used in parsing or decision-making, we monitor the number of tokens consumed and hence the API cost incurred for each session or task. Many advanced AI features (e.g., GPT-based parsing or reasoning) have usage-based pricing.

We compute metrics like tokens per order and even AI cost per task, which is an emerging KPI in agentic systems.

For instance, if processing a mobile activation averages 2,000 tokens of an LLM, we know the cost is, say, \$0.02 per activation (depending on model pricing).

This helps us optimize prompts and model calls — if a particular step is using an unusually high number of tokens, we investigate if the prompt can be streamlined or if a smaller model could be used.

We also set organizational budgets: e.g., no more than \$X per month on AI usage. The system can raise alerts or throttle certain non-critical LLM calls if the token usage approaches the limit. By tracking token consumption on a per-interaction and aggregate level, Cloudnet.ai ensures the **AI stays cost-effective**.

This metric also feeds into decisions about model deployment (for example, justifying a fine-tuned smaller model if it drastically cuts token usage with minimal accuracy loss).

We monitor how often the agent is calling various internal and external APIs, both to manage costs (some APIs might incur transaction fees) and to avoid hitting rate limits or causing performance bottlenecks.

For each type of tool call (e.g., CRM lookup, inventory check, credit score API), we track requests per minute/hour. This helps in capacity planning — ensuring backend systems can handle the AI's load — and in detecting inefficiencies (e.g., if the agent is calling the same API multiple times where one call would suffice).

If an API has a quota (say, 1000 calls/day), the platform can enforce that by queuing or rate-limiting calls once the quota is near.

Additionally, by analyzing call patterns, we might discover opportunities for caching. For instance, if in one customer session the agent calls the pricing API 5 separate times, that could be optimized to once if we cache the response.

Reducing redundant calls both improves latency and cuts unnecessary load. We define KPIs like average API calls per order and aim to minimize them without impacting functionality.

This ensures the solution scales efficiently — a telco with millions of subscribers must be confident that an AI agent won't spam their systems. By keeping API usage well-governed, we uphold telecom-grade operational discipline.



Request Success/Failure Trends:

The framework continuously analyzes trends in the success rate and failure rate of agent actions and external calls. A **request failure** could be an API call that returns an error, a tool invocation that times out, or a task the agent could not complete (which might contribute to order fallout as well). We aggregate these failures over time to see if there are rising trends.

For example, if the error rate for “Submit Order API” climbs from 0.5% to 5% in a day, that’s a red flag that something changed (perhaps an upstream system issue or a new bug).

The system is set to alert on such anomalies. Monitoring trends rather than just individual errors helps catch systemic issues early.

In many cases, the agent might retry on failures or use fallbacks, so the operation might still complete — but the trend metric would reveal a hidden problem (like increased latency or intermittent failures).

If certain error patterns occur repeatedly (e.g., many timeouts or specific error codes), the system can even trigger automated mitigations or rollbacks.

For instance, an increase in failures might automatically route new tasks to a backup process or switch the agent to a “safe mode” with minimal actions until the issue is resolved.

On the success side, we also watch for throughput trends: if the number of successful completions per hour suddenly drops, that could indicate a slowdown.

By keeping a pulse on success/failure trends, we maintain high reliability and can provide five-nines style assurances. This metric-driven vigilance is akin to how telco NOC teams monitor network KPIs — any unusual spike in failures pages the on-call team.

It ensures that if the AI agents begin to under-perform or encounter external system issues, we catch it immediately and control the situation, preventing runaway costs or customer impact.



In addition to these metrics, Cloudnet.ai employs **cost control levers** such as dynamic scaling and configuration tweaks.

For example, if token usage is spiking during peak hours, the system might route some interactions to a lighter ML model or turn off verbose logging to save costs.

Likewise, if certain complex requests consistently consume a lot of resources, we might choose to handle those via a scheduled batch process instead of real-time. The cost monitoring dashboard provides a view of dollars (or compute credits) spent per day on AI calls, broken down by agent and function, so product owners can measure ROI.

We also correlate cost metrics with business outcomes (e.g., cost per order vs. revenue per order) to ensure the automation is financially justified.

By keeping a close eye on resource usage, Cloudnet.ai guarantees that Agentic BSS scales in a **cost-efficient** manner. Telcos operate on thin margins and large volumes, so any AI solution must not only improve metrics like AHT and FCR but do so economically.

Our framework's cost controls give operators confidence that the AI's benefits (time saved, higher FCR, etc.) outweigh the operational costs — and if not, the metrics will make it evident, allowing adjustments. In short, just as we instrument for performance and safety, we instrument for cost, making **optimization and accountability** a continuous practice in the Agentic BSS deployment.



Continuous Improvement and Tuning:

The above metrics are not static report cards — they actively inform Cloudnet.ai's continuous improvement loop. Data from offline evaluations and production KPIs feeds back into retraining models, updating agent skills, and refining workflows on an ongoing basis.

For example, if parsing accuracy for a certain slot drops or new slang emerges in customer queries, those insights trigger an update to the NLU model.

If the containment rate isn't climbing as expected, we analyze the override cases to identify new automation opportunities.

Regular review of safety metrics (like override and violation incidents) helps tighten policies or expand the agent's knowledge to reduce false positives.

This closed-loop learning approach ensures the Agentic BSS becomes **smarter and more reliable over time**, essentially self-optimizing while remaining within governed bounds.

All improvements are tracked against the same evaluation framework to verify gains (e.g., a model update is only deployed if it shows higher parsing F1 and no regression in recommendation quality on the offline test set).

This disciplined, metrics-driven evolution aligns with enterprise telecom expectations — much like TM Forum's continuous improvement ethos — and assures that Agentic BSS keeps meeting **analyst-ready, telecom-grade standards** as it scales.

In essence, Cloudnet.ai treats each metric as a lever for tuning, ensuring Agentic BSS not only performs well today but keeps getting better in a measurable, auditable way.

Sources:

- 1 Auxiliobits — Evaluating Agentic AI in the Enterprise: Metrics, KPIs, and Benchmarks (instrumentation and continuous improvement best practices)
- 2 Arxiv (Telco AI Operating System) — Note on embedding health checks, telemetry, and audit trails in agent wrappers.
- 3 Fluid AI — The KPI Blueprint for Agentic AI Success (KPIs like completion rate, latency, and recommendation adoption)
- 4 Zendesk — Definition of Average Handle Time (AHT) and industry benchmarks.
- 5 RingCentral — Definition of First Contact Resolution rate (FCR).
- 6 Wikipedia (Telecom BSS) — Explanation of Order Fallout conditions in order processing.
- 7 Valens Ducis Medium — AI orchestration in telecom (on detecting error patterns and triggering alerts).

10

Implementing
Agentic BSS
in
Cloudnet.ai's
Product
Stack



How Agentic BSS Maps to Product Layers

Agentic BSS is delivered through a layered integration of Cloudnet.ai's products, ensuring AI capabilities augment — not replace — the existing BSS foundation.

The solution spans three key layers that work in tandem
(see Architecture Diagram: Order Genius Hub - Veris Agent Builder - Veris Suite):

Order Genius Hub:

This is the Copilot experience embedded in the CRM user interface. It serves as the front-end “agent” that interacts with users (e.g. customer service reps or even end-customers) via natural language.

Order Genius Hub understands intents and provides real-time assistance within the familiar CRM workflow.

For example, it can guide an agent through a new order by automatically populating forms and making recommendations, all within the CRM screen.

Veris Agent Builder:

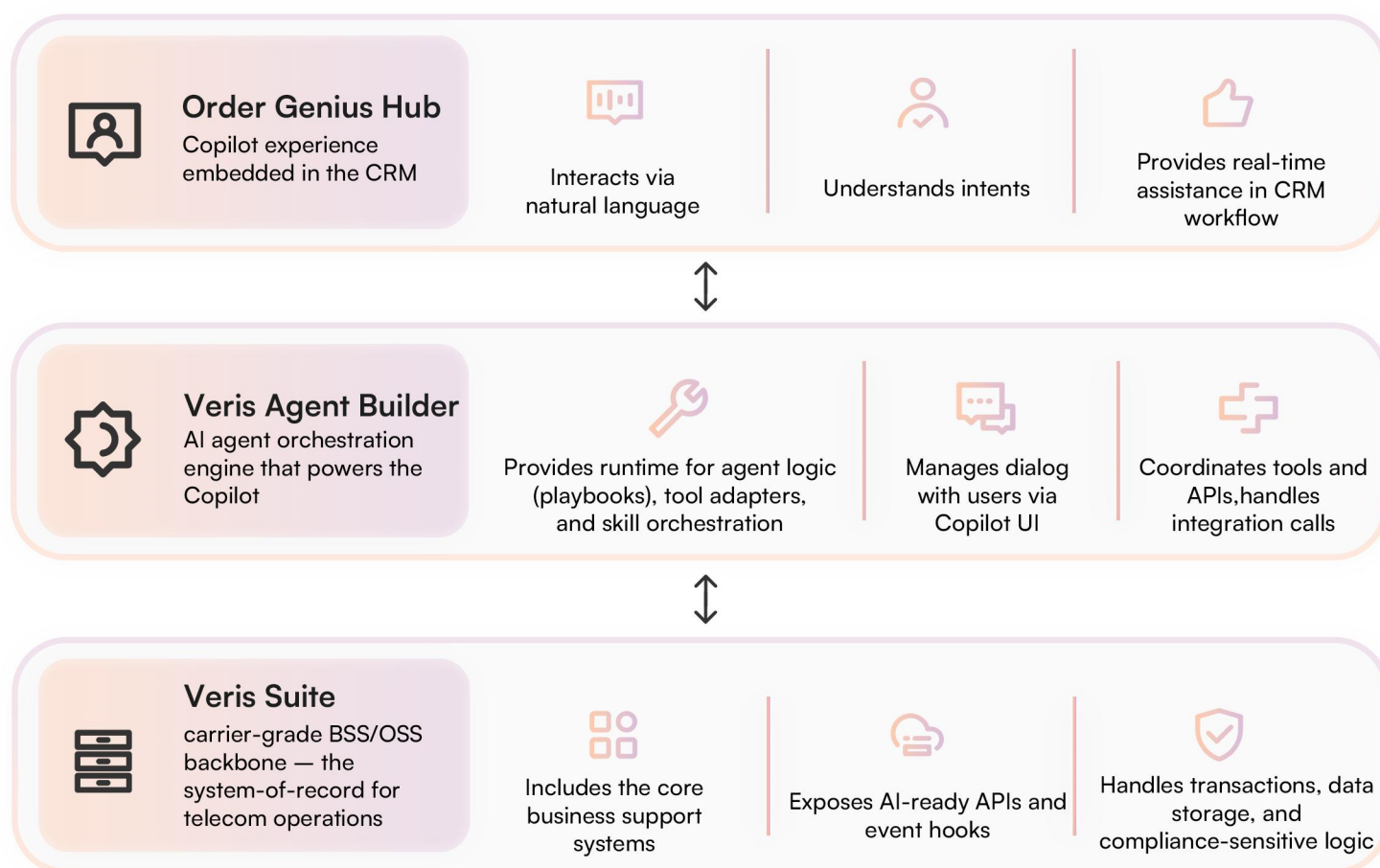
This middle layer is the AI agent orchestration engine that powers the Copilot. Veris Agent Builder provides the runtime for agent logic (playbooks), tool adapters, and skill orchestration. It is a low-code **agent builder** environment where Actions are composed into Skills, and Skills into full Agents.

In practice, Veris Agent Builder executes the playbook for a given intent — coordinating multiple specialized agents or tools, managing dialog with the user via the Copilot UI, and handling integration calls. It acts as the “brain” behind the Copilot, deciding which steps to take and which backend APIs to call.

Veris Suite:

This is Cloudnet.ai's carrier-grade BSS/OSS backbone — the system-of-record for telecom operations. Veris Suite includes the core business support systems (product catalog, order management, billing, etc.) and exposes **AI-ready APIs and event hooks** for external automation. Rather than being replaced, Veris Suite is actively leveraged by the agentic layer: it handles all final transactions, data storage, and compliance-sensitive logic. In the Agentic BSS architecture, Veris Suite provides the rock-solid foundation on which AI agents act.

All agent-initiated actions (e.g. creating an order, updating a billing record) ultimately flow through Veris Suite's governed processes. This ensures that while the BSS becomes more intelligent and proactive, it remains auditable and reliable.



By mapping the Agentic BSS vision onto these product layers, Cloudnet.ai ensures a clear separation of concerns. The Copilot (Order Genius Hub) delivers an intelligent UX at the channel level, Veris Agent Builder handles decision-making and orchestration of tasks, and Veris Suite sustains the heavy-duty BSS functions.

Importantly, the agentic capabilities build **on top of** the proven Veris Suite — accelerating innovation without disrupting existing telco-grade systems.



Supported Agent Patterns

Cloudnet.ai's implementation supports multiple **agent patterns** to cover a range of telco use cases.

Rather than a one-size-fits-all bot, the platform enables different types of agents optimized for specific interaction styles and tasks.

Key supported patterns include:

Single-skill agents:

Focused agents that perform one well-defined task or “skill” exceptionally well. For instance, an agent may specialize in **plan recommendations** — given a customer's usage profile, it invokes a pricing engine and returns the best plan.

These agents are modular and can be called upon as functions for larger workflows.

Intent router agents:

A top-level agent that interprets a user's query and routes it to the appropriate skill or sub-agent. In complex service scenarios, an intent router acts like a smart dispatcher — e.g. detecting whether a customer's message is about billing, technical support, or a new order, and then invoking the corresponding specialized agent or playbook.

This ensures users are handed off to the right process flow without manual triage.

Classifier copilots:

AI copilots that assist with classification or triage tasks in context. For example, a classifier copilot might analyze incoming trouble tickets or customer emails to categorize their issue (network problem, device issue, payment dispute, etc.), helping human agents prioritize and respond correctly.

This pattern uses the agent's language understanding to label or summarize information, typically working side-by-side with humans.

Long-task agents:

Agents capable of handling extended or asynchronous processes that unfold over time. Telecom operations often involve long-running steps (for instance, waiting for number porting to complete or monitoring a network fault resolution).

A long-task agent can initiate a process, pause or wait for external events, and then resume to complete the next step. It maintains state over hours or days as needed. Such agents are useful for **assurance** use cases or any scenario where the agent must “keep an eye” on something and act when conditions are met.



Workflow agents:

These agents orchestrate multi-step flows, essentially acting as an autonomous process manager. A workflow agent can execute a complex sequence of actions involving multiple tools and sub-agents to fulfill an end-to-end transaction.

For example, in an **order fulfillment** scenario, a workflow agent might handle a customer's request to upgrade service by coordinating a series of steps: checking eligibility, recommending a plan, provisioning the service, and updating billing—all following a defined playbook.

Workflow agents leverage the full power of the platform (dialog, branching logic, tool calls) to carry out structured processes with minimal human intervention.

Different patterns can be combined as well.

For instance, an intent router agent might first classify a request and then invoke a workflow agent which in turn uses several single-skill agents for specific tasks.

This modular approach reflects real operational needs — some AI agents serve as specialists, others as generalists or coordinators. Cloudnet.ai's Veris Agent Builder allows defining and managing all these agent types in a unified way.



Agent Pattern



Example Telco Use Case

Single-skill agents Plan recommendations

Intent router agents Billing/Support/New Order routing

Intent router agents Trouble ticket classification

Long-task agents Number porting/fault monitoring

Workflow agents Multi-step order activation

The screenshot displays a user interface for managing a mobile service order. It includes sections for Order Items, Delivery Information, and Additional Information, with various input fields and buttons for actions like 'Change', 'Reset', and 'Submit'.

Order Items

- Number Information: 88882133 (Port Number, New Number, Change)
- Card Type: Sim 311 Voice | Number Fee: 80.00 | Porting Date: -- | Service Operator: --
- Plan Information: Business Mobile 25 GB (RC 129.00/MO, OTC 120.00, Change)
- Subscription Commitment: 12 months | New Phone Commitment: 12 | Phonebook STATUS: Normal number | telefonbog
- Device Information: Apple iPhone 17 Pro 256GB Blue (OTC 8,099.00, Change)
- Mobile Insurance: Screen Insurance (Contact Number: 33215500 | Email: Pandora@Pandora.dk | Policy Number: FOR105155002, RC)

Delivery Information

- Pick up in store | Delivery to address
- Storegade 13 3720 Aakirkeby
- Carrier: PostNord 02-Parcel
- Receiver: Pandora Attention People: --
- Contact info: 33215500 Pandora@Pandora.dk

Additional Information

- Pay Monthly: 129.00/MO
- OTC Price: 8,179.00
- Buttons: Reset, Submit

Right Panel:

- All basic validations passed
- Customer Name: Pandora(23210320135423)
- Recommended Numbers: 88882133, 88882833, 88882944
- Recommended Offers: Business Mobile 25 GB (129.00/MO), Apple iPhone 17 Pro 256GB Blue (8,099.00), Business Mobile 50 GB (229.00/MO), Apple iPhone 15 128GB Black (3,199.00), Business Mobile 25 GB (129.00/MO), Apple iPhone 15 128GB Black (3,499.00)
- Recommended Insurance: Mobile phone Insurance (89.00/MO), Screen replacement (39.00/MO), Screen Insurance (0.00/MO)
- Type a message...



Playbooks as the Control Layer

In Cloudnet.ai's architecture, **playbooks** serve as the control layer that defines each agent's behavior.

A playbook is a configuration (no-code/low-code logic) that encapsulates the workflow for a given intent or process, and it can be updated without changing application code.

Playbooks typically define several key elements of an agent's operation:

Intent triggers:

The conditions or events that invoke the playbook. Triggers can be user intents (e.g. NLP-detected intent like "upgrade plan"), specific keywords, or backend events.

The playbook listens for these triggers to decide when to launch an agent-driven process.

Required slots:

The critical pieces of information (parameters) that must be collected to fulfill the task. For example, for an order-related playbook, required slots might include the customer ID, desired product, address, etc.

The playbook can specify which slots are mandatory and prompt the Copilot to gather any missing info from the user.

Steps and flows:

The sequence of actions the agent will take, defined in a mix of dialog steps and tool/agent calls. This can include asking questions to fill slots, invoking external APIs (tools) to get data or perform actions, calling subordinate agents/skills, and branching based on results.

The playbook acts like a script or flowchart governing the agent's decisions and responses at each step.



● **Fallbacks and error handling:**

Definitions of what the agent should do if something goes wrong or if the user's input is not understood.

Playbooks can include fallback prompts (e.g. rephrase a question, escalate to a human agent) and timeouts or default actions if an expected event doesn't occur. This ensures the agent can gracefully handle exceptions without derailing the user experience.

● **Governance mode:**

Settings that determine how the agent's outputs are treated in production — for instance, whether the agent is in “**draft**” mode (human-in-the-loop) or “**autonomous**” mode for a given action.

The playbook can enforce that certain steps only produce recommendations (requiring human confirmation) versus allowing an agent to directly commit changes if confidence is high and policy allows.

Because playbooks are configuration-driven, **updating an agent's behavior or business rules is as simple as editing the playbook** rather than writing new code.

This separation of logic from code accelerates iteration and allows business owners and analysts to refine AI-driven processes quickly. Moreover, the platform supports versioning of playbooks and **controlled rollout** of new versions. Teams can test a new playbook version in staging or with a small subset of live users, and gradually increase deployment as confidence grows.

This partial rollout and easy rollback capability provides an extra safety net, ensuring that updates to agent logic can be deployed in a measured, reversible way — critical for production governance in telecom environments.



Tool Adapter Layer and Integration Posture

To connect AI agents with the myriad of underlying systems in a telecom BSS/OSS environment, Cloudnet.ai employs a **Tool Adapter layer**.

Tool adapters abstract away the complexity of legacy system integration, providing a standardized interface for the agents to invoke external functions.

Instead of the agent needing to know how to call a specific CRM SOAP API versus a billing REST API, it calls a tool adapter in a uniform way and the adapter handles the details. This design allows the Agentic BSS to plug into existing IT landscapes with minimal friction.

Key integration modes supported include:

API Integration:

The adapter can call RESTful or SOAP APIs exposed by BSS/OSS components (or even third-party services). Each tool adapter is configured with the endpoint details, payload schemas, and authentication needed, so that an agent step can simply request “invoke tool X” and get the response.

This standard API invocation mechanism covers integration with modern systems (e.g. REST/JSON endpoints) as well as older interfaces, without changing the agent logic.

Event-Driven Triggers:

In addition to direct API calls, Veris Agent Builder can react to events from the Veris Suite or other systems. For example, an event like “order completed” or “payment failed” can automatically trigger an agent (via a playbook) to take follow-up actions.

Tool adapters can subscribe to message queues or webhook events, translating external events into playbook triggers.

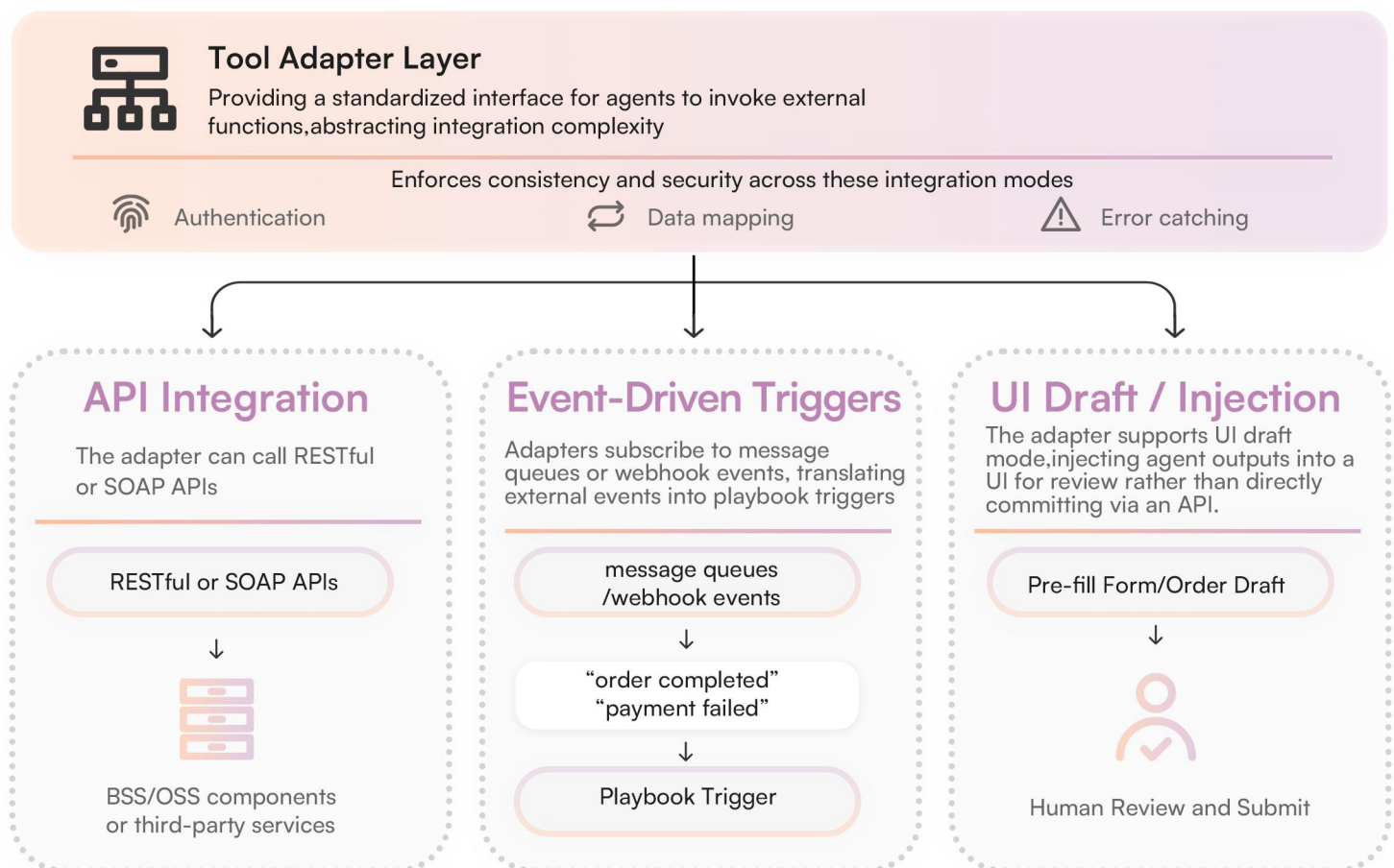
This event-driven posture ensures the Agentic BSS isn’t just reactive to user commands, but can also proactively respond to system events (e.g. auto-create a support ticket if a billing anomaly is detected).

UI Draft / Injection:

Not all legacy systems support APIs for every function, and sometimes human oversight is required by design. For such cases, the tool adapter layer supports a **UI draft mode** — where the agent produces an output that is injected into a user interface for review rather than directly committed via an API.

For instance, the agent might pre-fill a form in the CRM or generate an “order draft” that a human can verify and submit. This approach leverages existing UIs and RPA-style interactions to integrate where APIs are not available, while keeping a human in the loop.

The Copilot (Order Genius Hub) is instrumental here, showing the AI’s drafted inputs to the user for quick confirmation.



Across these integration modes, the Tool Adapter layer enforces consistency and security — handling authentication, data mapping, and error catching uniformly.

It enables Cloudnet.ai’s agents to work with a wide range of operator systems without bespoke code for each integration, accelerating deployment in complex IT environments.



Governance in Production

Running autonomous agents in a live telecom environment demands strong governance to maintain trust and compliance.

Cloudnet.ai's Agentic BSS adopts a “draft-first” deployment philosophy to ensure a safe introduction of AI into operations. In practice, this means initial agent outputs are treated as recommendations or drafts,

requiring human confirmation before final execution:

Draft-First Execution:

By default, AI agents in production generate draft actions — for example, a Copilot might draft an order change or a recommended solution — which is then reviewed by a human (CSR, manager, etc.) before being officially submitted. This human-in-the-loop stage guarantees that no critical transaction happens without oversight when the AI is still gaining trust. Over time, as the agent's accuracy is proven, specific tasks can be allowed to auto-complete.

Override Thresholds & Escalation:

The system provides configurable confidence thresholds and business rules to decide when an agent can act autonomously. If an agent's confidence score on a task is below a threshold, or if it encounters an unfamiliar situation, it will automatically escalate to a human operator or fall back to a safe default. Conversely, for routine low-risk tasks, if confidence is very high and past performance has been reliable, the agent may be permitted to complete the action without intervention. These guardrails prevent the AI from going beyond its bounds and allow telecom operators to dial up or down the level of autonomy gradually.

Audit Trails and Traceability:

Every agent session and action is logged in detail (with session IDs, timestamps, inputs/outputs, and decision pathways). This creates a robust audit trail for compliance and troubleshooting. If a suggestion was made by the AI and approved by a human, the record shows both the AI's recommendation and the human's confirmation. If the AI took an autonomous action, the log captures why (e.g. confidence level, playbook version) and how. Such trace logs enable teams to conduct reviews, explain outcomes to regulators, and continuously improve the agents. Governance dashboards in Veris Agent Builder allow monitoring of agent performance, override rates, and any escalations, providing full transparency of AI operations.



Safe Progression of Automation:

Cloudnet.ai recommends a phased approach to increasing AI autonomy. Initially, agents operate in a **co-pilot mode** (assistive/draft-only). As confidence grows, operators can expand the agent's scope — e.g. allowing it to auto-complete certain steps or handle certain intents end-to-end during off-peak hours.

Eventually, a mature agent might handle a significant fraction of tasks autonomously, with humans focusing only on exceptions. This gradual progression ensures that the organization, processes, and AI learn and adapt together, minimizing shocks.

Governance policies can be set to always keep a human fail-safe for critical actions, ensuring the system remains **telco-grade in reliability and compliance** even as more processes become AI-driven.

Deployment Posture

Cloudnet.ai's Agentic BSS components (the Veris Agent Builder engine and Order Genius Hub Copilot modules) are designed with cloud-native principles, offering flexibility in deployment.

They can be deployed on public cloud infrastructure, on-premises within the operator's data center, or in hybrid models — depending on the operator's regulatory and latency requirements.

This flexible deployment posture means a telecom can benefit from Agentic BSS whether they prefer a fully managed SaaS in the public cloud or need to keep the solution in-house for data sovereignty. By default, the solution is containerized and Kubernetes-ready, enabling scalable and resilient operations.

Ultimately, Cloudnet.ai aligns with the operator's IT strategy: delivering the Agentic BSS vision in the environment of choice without compromising on performance or security.

Competitive Landscape

11

The following capability matrix contrasts “AI add-on” approaches (incremental copilot/assistant layers) with fully **agentic BSS** architectures (integrated, autonomous execution layers).

It is based on industry frameworks and analyses (e.g., TM Forum’s ODA principles and Gartner’s agentic AI research):

 Capability Dimension	 AI Add-ons (Assistive Layer)	 Agentic Architecture (Integrated Execution Layer)
Scope of Automation	Limited to point solutions or channels (e.g. chatbot guidance, summarization, recommendations)	End-to-end orchestration of complex workflows across BSS/OSS domains (catalog, order management, provisioning, billing, etc.)
Integration Depth	Overlay on existing siloed systems via UI plugins or narrow APIs	Deep integration with system-of-record via open APIs/events; componentized per TM Forum ODA guidelines
Governance Model	Human-in-the-loop by default; AI as assistant with no end-to-end audit trail	Built-in governance (draft→validate→approve→commit), full audit logs, policy enforcement, and explicit human gates for sensitive actions
Exception Handling	Largely manual fallback with human operators untangling errors	Automated diagnostics and closed-loop remediation playbooks detect and fix exceptions across domains
Scalability & Flexibility	Incremental gains; scaling limited to specific tasks or channels	Designed for multi-agent orchestration at scale, with policy-driven autonomy across domains
Operational Impact	Improves localized productivity (e.g. smaller cost reductions, speedups)	Drives measurable improvements in KPI targets (reduced AHT, higher FCR, lower order fallout, lower cost-to-serve) across operations

This comparison synthesises insights from standards and analyst research (for example, TM Forum’s vision of modular, API-led BSS/OSS [referencing [TM Forum ODA] for simplifying and automating operations] and Gartner’s definitions distinguishing narrow assistants from autonomous agents).

In sum, AI add-ons can ease tasks in isolation, **but an agentic BSS is needed for strategic, telecom-grade autonomy that meets the complexity and scale of modern networks.**

12

Conclusion & Appendices

Conclusion:

Agentic BSS represents a **transformative evolution** in telecom operations.

By moving beyond isolated assistants to a governed, auditable execution layer, operators can tackle 5G/IoT complexity and aggressive SLAs while controlling costs and improving customer experience.

Cloudnet.ai's layered solution-

- **Veris Suite** (system-of-record)
 - **Veris Agent Builder** (AI orchestration/agents)
 - **Order Genius Hub** (copilot interface)
- embodies this approach, balancing rapid automation with built-in safety and accountability.

This whitepaper has provided a rigorous foundation for understanding Agentic BSS:

its architecture principles, value metrics, governance framework, and risk mitigations. Analysts and operators can use these insights to evaluate “agentic” initiatives objectively, plan phased rollouts, and align them with business KPIs and compliance requirements.

(TM Forum's ODA and other industry frameworks underscore that such an integrated, policy-driven architecture is essential for future telecom agility.)



Appendix A: Glossary

Agentic BSS: An AI-driven operating model where intelligent agents execute and optimize end-to-end BSS workflows under full governance.

AI-to-AI Operations: Coordination among multiple AI components (agents, models, tools) to autonomously manage complex tasks.

Playbook: A workflow definition (often low-code) that sequences agent actions, validation checks, and governance steps.

Tool (in Agentic BSS): A backend API or service (e.g. check eligibility, reserve inventory) that an agent invokes to perform an action.

Draft Mode: A governance mechanism where agent-generated transactions are proposed as drafts, requiring human review/approval before commitment.

Human-in-the-Loop (HITL): The practice of keeping a human operator in the decision chain for approval or correction, ensuring oversight.

Order Fallout: An order or task that fails or is blocked during automated execution, requiring manual intervention to complete.

Average Handling Time (AHT): Standard metric for the average time to complete a customer transaction (including talk, hold, and after-call work).

First Contact Resolution (FCR): The rate at which customer issues or requests are resolved in the first interaction, without repeat contacts.

References:

- ① TM Forum Open Digital Architecture (ODA) briefs;
- ② Gartner agentic AI research; National Institute of Standards and Technology (NIST) AI Risk Management Framework;
- ③ PwC Global Telecom Outlook 2024—2028;
- ⑤ McKinsey & Company on scaling AI in telecom.

**Agentic BSS
is not a vision,
it's ready for
production
today.**