KaiROS™

BRIEFING · FOR DATA CENTRE OPERATIONS AND TECHNOLOGY LEADERSHIP

Beyond the Network: Intent-Based Operations for the Complete Data Centre

What it is, why infrastructure leadership must move to it, and the technology gap that kept it out of reach until now.

What is Intent-Based Operations?

Data centre networks run on complex spine-leaf underlay and overlay architectures such as VXLAN-EVPN. Managing that fabric manually, box by box through the CLI, is slow and error-prone. Intent-Based Operations (IBO) — in the data centre, Intent-Based Networking — replaces that with a declarative model: instead of configuring VLANs, VRFs, and BGP peering on dozens of individual switches, the operator declares the desired state, and the controller generates and deploys the entire configuration, then continuously checks live telemetry against that intent to confirm the network stays as declared.

Why operators need to move to it. Three forces are converging, and none of them can be met by manual, box-by-box operations:

Machine-speed change

Automation and AI agents propose and execute changes far faster than any approval process built for people.

A growing rulebook

Safety, tenant SLAs, compliance, and sustainability obligations keep expanding — enforced today by experienced people paying attention.

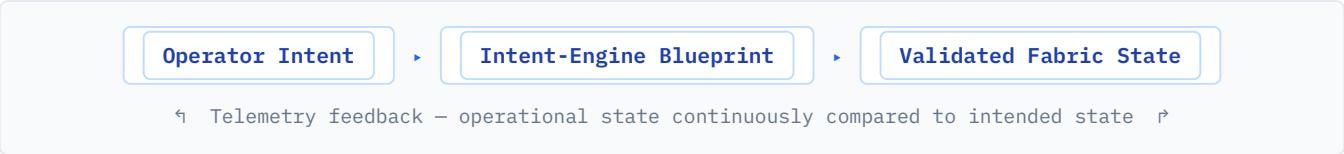
Proof on demand

Auditors and regulators increasingly require operators to show who approved an action, on which facts, under which rule.

Declaring intent once and letting a system enforce it is the only model that meets all three. That is why data centre operations has to move to IBO.

The model is proven where it has been applied — which, so far, is only the network. Provisioning a multi-tier application with its firewall rules used to take days; with intent-based networking it takes minutes, with far less room for human error.

The defining capability is continuous verification against a single source of truth: a blueprint of how the fabric should behave, continuously compared against the operational state reported by telemetry.



A dropped route, a wrong port, or a blocking ACL is flagged the moment it drifts. On this foundation, controllers run real operations:

Automated tenant isolation and micro-segmentation.
To isolate a high-security suite for compliance, the administrator declares *"isolate Zone X from all zones except Secure Gateway Y,"* and the engine builds the VNIs, route targets, gateways, and security tables — no manual syntax error that could leak data across racks.

Predictive congestion management.
When an AI-training cluster starts dropping packets on a leaf-spine link, the system detects the breach of *"ensure zero packet drop for AI-Cluster-Alpha"* and recalculates ECMP weights to drain traffic onto idle paths before performance degrades.

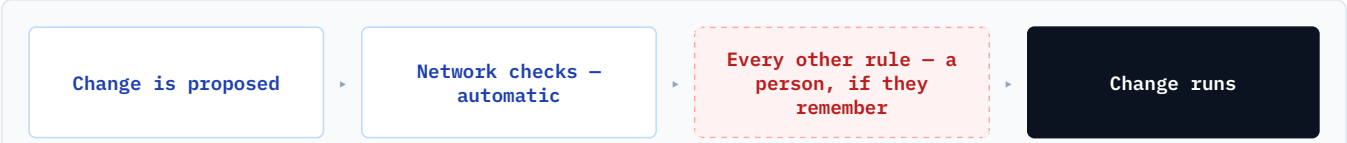
Pre-flight change simulation and rollback.
Before a policy update touches thousands of VMs, the engine simulates it, checks for conflicts with existing rules or routing, and deploys only if it passes, with one-click rollback afterwards.
So far, all of this exists only for the network. The rest of this brief is about everything else the operator is accountable for.

Why full IBO has been out of reach

The network is the only part of the operation where intent is executable today: fabric automation deploys the operator's network intent and continuously verifies the fabric matches it.

But a data centre is governed by a second set of rules the controller never sees, because they are not part of the network at all: power and cooling state and N+1 commitments; permits-to-work; change freezes; contractual SLAs; compliance boundaries. Each states what must *always* be true, and none of it is checked automatically — it lives in documents and contracts, enforced by attention.

The network's rules are checked by software. Every other rule the operator lives under is checked by people — or not at all.



Power, safety, freezes, SLAs, compliance — checked by attention, not by the system. That is where the gap is.

What this looks like in practice

From day one, KaiROS — Reshuffle.AI's decision layer for the data centre — is a decision support system, and we use the term deliberately. It does not take decisions from engineers; it gives every decision its facts, its rules, and its proof — the operator's own rulebook, in two modes:

QUICK QUESTION

"What is the best time to do the UPS upgrade?"

Two windows survive — and the engineer never had to say what the constraints were. The system works out which rules the upgrade touches, including the ones nobody mentioned: affected tenants must be notified — it finds who, and how much notice each contract requires, holiday-adjusted; the change-freeze calendar; other scheduled work that stacks risk; the redundancy rules barring concurrent module work. Each window is kept or eliminated, with its reason:

Oct 3–5	Oct 10–12	Oct 17–19	Oct 24–26	Oct 31–Nov 2	Nov 7–9
TENANT NOTICE	CHANGE FREEZE	OPTION A	CRAH 2 WORK	HOLIDAY WEEK	OPTION B

Take Option A, and tenant notices are due by 3 October — drafted, per tenant, ready to send.

Why RAG, GraphRAG, or an AI agent can't do this as well. Retrieval — plain or graph — finds documents, and this answer is in none of them: it is calculated across contracts, calendars, and schedules together. An AI agent with a workflow can fetch the calendars and calculate against predetermined constraints — but someone must tell it the constraints, and the model driving it can still miss one, or invent one, because that is what hallucination is; and it cannot tell you what it missed. Ours cannot hallucinate on the decision path: the rules come from your own documents, checked by your engineers — every rule that applies is found, none invented, and the same question always returns the same answer, with the working shown.

GUIDED ANALYSIS — PLACEMENT

"A customer wants to deploy a new AI inference service — twelve racks, 40 kW per rack, liquid-assisted cooling, N+1 power, go-live in two weeks. Where can we place it?"

The system reads the current state from the systems that already hold it: power headroom per row (EPMS), cooling capacity (BMS), rack and floor space (DCIM), what each row's contract promises, and every change already scheduled in the next two weeks — the checks an engineer would spend a day assembling by hand, gathered in one pass.

It asks for what no system holds. "Does the service need low-latency access to the carrier interconnect? Split across two rows acceptable? Is the two-week date fixed?" The engineer answers; anything unanswered holds the analysis — it never guesses. That restraint is the point: an assistant that fills gaps with assumptions creates risk; one that stops and asks can be trusted.

It narrows the field, with reasons. Of the candidate rows: rows 3–5 fail the cooling class — air-only. Row 13 fails on something that hasn't happened yet: with Thursday's UPS module out, this load would push the plant past what N can carry — on rows sold as N+1. Row 8 clears once its B-feed repair (ticketed, Monday ETA) completes. Rows 21–22 pass every check, including the split. Row 13 is the catch that matters: a placement that looks fine on every dashboard today would have broken a contractual guarantee on Thursday — found before anything was approved.

It answers with the working. Rows 21–22 now; Row 8 from Monday — each option showing the rules it passed and the facts used — change request pre-filled, ready for approval. The engineer decides, spending judgment on choosing between proven options rather than hunting for hidden conflicts.

The value is not only the blocked mistake: capacity gets sold instead of stranded, because the margin is calculated, not guessed.

Guided analysis, end to end. The path is derived per situation — not authored in advance.

Why a workflow system can't do this. A workflow runs the steps someone built into it — it checks what its designer thought of. Nobody's flowchart contains "join this request to another team's UPS work and re-check the N+1 promise," and every new rule means a developer writing another step. A workflow also doesn't weigh facts: when a check times out it takes the error path — or approves when the clock runs out — where here a missing fact simply holds the answer. And its audit trail shows that the process ran, not why the action was allowed. Here the path is worked out fresh from the rules and the facts, every time.

And it fits the stack the operator already runs. Facts in, read-only, from DCIM, EPMS, BMS, ITSM, GRC, and the fabric controller — and from predictive analytics and AIOps too: a cooling forecast or an anomaly alert arrives as one more fact for the rules to weigh. Decisions out through the same stack: the change request lands in the operator's own ITSM, notices go through existing channels, every verdict writes an audit record. Nothing is replaced — the layer decides; the stack executes.

The same mode also runs backwards:

GUIDED ANALYSIS — DIAGNOSTICS

"Row 12 dropped to N at 02:14 last night — what happened?"

It gathers what the rules need from records that already exist: the power-quality log shows a utility dip at 02:14; the EPMS shows the ATS transfer it forced; the change calendar shows module 3 out for planned maintenance — so from 02:14 until the module returned at 02:54, the plant stood at N. Every system did its job — which is why no alarm marks the part that matters: obligations do not trip sensors.

Then it raises the question the engineer didn't ask. Under Tenant K's contract, a redundancy loss over thirty minutes is a reportable event — notice due within 24 hours, an SLA credit triggered.

And it asks the one question that decides it. The clause exempts losses during maintenance notified to the tenant in advance — and no notice to Tenant K is on record. "Was Tenant K notified of the module-3 work?" **No** → reportable: notice due by 02:14 tomorrow, draft written, credit computed, exposure stated. **Yes** → the exemption applies: attach the notice, and nothing is owed. One fact, two different obligations — so it asks, rather than guesses. Get it wrong one way and a contractual notice is missed; the other way, a credit is paid that was never owed.

By 03:00 the incident file is assembled either way — what happened, why the plant was at N, and what is owed, or why nothing is. Root-cause tools stop at what happened; this continues to what follows. The engineer answers, and decides what goes out.

Quick answers are grounded in the compiled rulebook; computations — deadlines, thresholds — and all of guided analysis run on the deterministic path.

This is the industry's own data, not a scenario. In Uptime Institute's 2025 Annual Outage Analysis, nearly 40% of organisations report a major human-error outage within three years — and 85% of those incidents come down to staff not following procedures, or the procedures themselves being wrong. The rules exist. Enforcement is a person with a checklist. Meanwhile AI clusters raise the rate and cost of a wrong change, and auditors and regulators require proof of who approved an action, on which facts, under which rule — under the EU AI Act and NIS2, a requirement, not a preference.

What is neuro-symbolic AI?

Most of the AI in the headlines is one kind — neural AI: the large language models such as ChatGPT. They are extraordinary at language, and they work by prediction. Given a question, a model predicts the most likely words in response. That makes them fluent and flexible, but it also means the same question can produce different answers, they cannot show a checkable line of reasoning, and they will always produce *an* answer even when they should not — which is what people mean by hallucination.

There is an older, opposite kind of AI: symbolic systems. These do not predict; they follow explicit rules to a logical conclusion, the way a calculator or a tax engine does. Given the same inputs, a symbolic system always returns the same answer, and it can show every step. It is rigorous and auditable — but on its own it is rigid, and it cannot read a policy document or a contract written in plain English.

Neuro-symbolic AI combines the two, and keeps them in their proper roles. The language model does what it is good at — reading documents and turning them into precise rules, which people then verify — and the symbolic engine does what it is good at: making the actual decisions by following those rules, deterministically, with every step shown. The language model reads and explains; it never decides. Gartner's Hype Cycle for Artificial Intelligence, 2025 profiles neuro-symbolic AI as a form of composite AI that, in Gartner's words, "can augment and automate decision making with less risk of unintended consequences" — and the research literature calls it "the third wave" of AI (Garcez & Lamb, 2023), after symbolic systems and today's statistical models. Neither kind of AI is sufficient alone for work that has to be both understood from documents and decided with rigour.

Neural AI (ChatGPT, etc.)	Symbolic AI	Neuro-symbolic AI
Predicts the most likely words	Follows explicit rules to a conclusion	Neural reads and explains; symbolic decides
✓ Natural language understanding	✗ Natural language understanding	✓ Natural language understanding
✗ Deductive reasoning — poor	✓ Deductive reasoning — deterministic	✓ Deductive reasoning — deterministic
✗ Same facts, same answer	✓ Same facts, same answer	✓ Same facts, same answer
✗ Shows its working	✓ Shows its working	✓ Shows its working
✗ Hallucination — inherent	✓ Cannot hallucinate	✓ Hallucination-free decision path

Same five capabilities, compared like for like. The neural side handles the words; the symbolic side handles the judgement.

Why it is the technology that makes IBO possible

Why the fabric vendors cannot simply add this. Two reasons, both structural. Blueprint validators are closed-world shape-checkers: they test a change against a pre-enumerated model of the network. The rulebook requires the opposite discipline — deduction over the operator's obligations — where a missing fact is treated as unknown, never assumed fine — joined to live facts from systems the fabric controller does not own (DCIM, safety, GRC, contracts), with declared precedence when rules collide and honest abstention when facts are missing. That is a different kind of system, not a feature added to a validator. And this layer must be *neutral*: it governs actions from every proposer — the fabric controller, the DCIM automation, the human, the AI agent — and reads state from every source. A fabric vendor cannot be that layer, because it cannot sit in judgment over competitors' controllers or subordinate its own remediation loop to a check above it.

Neuro-symbolic AI is the architecture that fits. Reading obligations out of documents is language work, which the neural side handles; deciding an action against them with rigour and a proof is logic work, which the symbolic side handles. It is the gap in today's tooling: LLM agents are flexible but unreliable, runbook and blueprint automation is reliable but rigid — this is both. No single-technology approach can do it, and that, more than anything, is why full IBO has had to wait.

What the decision layer must do

Making IBO implementable means meeting the same bar network automation already meets for the fabric — a single compiled source of truth, continuous verification, determinism — plus what obligations add that network models do not have: facts spread across systems, rules that conflict, unknowns that must be admitted, and decisions that must be auditable. Concretely, the decision layer must:

1. **Hold the whole rulebook as one source of truth** — the operator's obligations compiled from their own documents and verified by their experts, the way the blueprint is the authoritative version of the network.
2. **Check across systems, not one model** — evaluate an action against network, power, safety, and contractual facts together, because the failures live in the combination.
3. **Be deterministic** — the same facts must always give the same result, with no variation by phrasing, sampling, or load.
4. **Produce an auditable result every time** — every decision, approvals included, carrying the rule applied, the facts and their sources, and the reasoning, re-checkable independently.
5. **Handle the unknown honestly** — if a fact is missing, stale, or unreachable, hold the action and say so; where a rule needs human judgment, route it rather than guess.
6. **Resolve conflicts by declared precedence** — an energy-saving policy that would shed a redundant feed against a tenant's contractual N+1; a freeze against an urgent remediation — deterministically, visibly, and surfaced before anything deploys.
7. **Run pre-flight and in-line, at change-management timescales** — in seconds to minutes, before any change reaches production.

REQUIREMENT	RUNBOOK AUTOMATION	BLUEPRINT VALIDATION	LLM AGENTS	NEURO-SYMBOLIC (REASONEX)
1 Holds the whole rulebook	Only what's hand-coded	The network model only	Reads it, doesn't hold it	Compiled & expert-verified
2 Checks across systems	Only where anticipated	Fabric-internal only	Imitated, not derived	Core capability
3 Same facts, same answer	Yes	Yes	Varies run to run	Yes — by construction
4 Proof for every decision	Execution log only	Validation report	Confidence score	Re-checkable record
5 Knows when to stop	Only if encoded	Closed-world — absent looks fine	Answers regardless	Holds, names the gap
6 Conflict & precedence	None	Network intents only	Weak	Declared hierarchy, shown
7 Pre-flight, operational speed	Yes	Yes	Fast but inconsistent	Yes

Categories, not vendors — where another approach is genuinely strong, the table says so. Only one column meets all seven.

No single conventional technology meets all seven. The sharpest evidence is on the language models: a 2025 Stanford–Yale study in the *Journal of Empirical Legal Studies*, testing the leading commercial legal-AI platforms, found they answered incorrectly 17–33% of the time. That is measured in a domain where a wrong answer costs rework and delay; infrastructure operations, where a wrong answer costs uptime, can tolerate far less. Only a neuro-symbolic engine — language to read the rules, symbolic logic to decide by them — meets all seven at once.

That engine exists: Reasonex, Reshuffle.AI's neuro-symbolic reasoning engine. It differs from the approaches in the table by construction, not by tuning. The rulebook is turned into machine-checkable rules once, verified by the operator's own engineers before it ever serves — not rediscovered per question by a model. Every fact is typed and carries its source and timestamp, and can be true, false, or unknown — and unknown never approves. Decisions are computed by a deterministic symbolic core the language model cannot touch, so the same facts always return the same answer, with a proof anyone can re-check. It was hardened in law — the most demanding rulebook domain there is — before being pointed at infrastructure.

Why Reasonex meets these requirements

Built on that division of labour — language reads and drafts, the operator's experts verify, a deterministic core decides — Reasonex meets each requirement directly:

Requirement 1	The operator's rules are compiled once into a single authoritative, versioned rule set, verified with the operator's engineers. When a contract, SOP, or policy changes, the neural component ingests the new text and drafts the rule updates for expert sign-off.
Requirement 2	Each proposed action is evaluated against that rule set together with live facts from the systems the operator already runs — DCIM, BMS, fabric controllers, ITSM, GRC, inventory — evaluated together, not one system at a time.
Requirement 3	The symbolic core is deterministic: identical facts always produce the identical decision, reproducible exactly in an audit.
Requirement 4	Every decision, approvals included, emits a record that can be re-checked independently — the rule, the facts with their sources, and the reasoning.
Requirement 5	Missing, stale, or unreachable facts hold the action and name what is needed; rules marked as judgment route to a named role with the reasoning assembled. It never guesses.
Requirement 6	Conflicts are resolved by the operator's declared hierarchy, and the resolution is shown, not silent.
Requirement 7	It is designed to run pre-flight and in-line at change-management timescales — seconds to minutes, before any change reaches production.

Delivered as **KaiROS™**, Reasonex is designed to sit alongside the fabric controller, not in place of it — read-only connections, and it never executes changes; the operator's own tooling executes behind the decision, drawing only on a closed, operator-approved action catalogue, so the system cannot invent an action. Each proposed action — from an engineer, a fabric controller, or an AI agent — returns one of four outcomes: **Act** — approved, with the proof; **Ask** — held, naming the missing fact; **Escalate** — routed to the responsible role; or **Stop** — blocked, with the violated rule and its source. Consultation — the two decision-support modes — needs no autonomy; from there it is earned per rule class, at the operator's threshold:

LEVEL 1	LEVEL 2	LEVEL 3	LEVEL 4	LEVEL 5
Observe	**Recommend**	**Approve to enforce**	**Enforce within limits**	**Enforce proven rules**
Decides and records; blocks nothing.	Proposes; a person applies.	Applies on sign-off.	Bounded blast radius.	Earned, per rule class.

Execution stays with the operator's tooling at every level, and the audited override for live incidents remains throughout.

Where to start. Consultation first: one hall, the operator's own documents, rules verified by the operator's engineers — live in weeks. Observation and enforcement are later settings, not preconditions.

The same Reasonex engine already runs in production, reasoning over compiled civil-procedure law across three jurisdictions as our product **MikeROS™** — a domain where a wrong answer carries professional sanctions. The same symbolic core evaluates rules, dependencies, and constraints in both — a clause in a statute or an N+1 threshold read from live telemetry. Only the rulebook and the facts change.

Network automation made one kind of intent executable. Reasonex makes the rest executable — turning Intent-Based Operations from a direction into something an operator can implement.

ABOUT RESHUFFLE.AI

Reshuffle.AI Pte. Ltd. is Singapore's first neuro-symbolic AI company. CSO: Prof. Ernest Chong (SUTD; originator of Algebraic Machine Reasoning, CVPR 2023). Selected into SUTD's ARISE programme.

reshuffleai.com/kairos **contact@reshuffleai.com**