Ø CONSTELLATION

The Operating System for Space Operations

The architecture of the space industry has changed. This paper is about the layer that change makes inevitable: mission control that turns intent into operations across sovereign, commercial, and federated infrastructure.

CONSTELLATION SPACE CORP

TECHNICAL WHITEPAPER · JULY 2026 · PUBLIC

CONTENTS

01

Why we built Constellation

Every satellite operator we have ever worked with runs the most advanced machines humanity has built — with spreadsheets, tribal knowledge, and heroics.

The spacecraft are extraordinary. The operations behind them are not. Contact schedules are negotiated over email. Ground stations are selected by habit. Weather is discovered when the downlink degrades. Every new ground provider is a months-long integration project, and every fleet of mixed vendors means another one-off telemetry pipeline maintained by whoever wrote it.

The people doing this work are exceptional. The tooling is what failed them: it was built for an era of one operator, one fleet, one ground network, and a handful of passes per day. That era is over.

We built Constellation because the coordination layer above spacecraft — mission control, in the oldest and best sense of the phrase — deserved to be engineered with the same rigor as the spacecraft themselves. We believe operators should define *mission intent*, and software should translate that intent into executable, auditable operations. Everything in this paper follows from that belief.

02

The next space economy

The limiting factor of the space economy will no longer be launch. It will be operational throughput — and the next trillion dollars of space infrastructure will be constrained by coordination, not hardware.

Every decade, the bottleneck moved:

ERA	THE BINDING CONSTRAINT
1960s	Reaching orbit at all
1990s	Launch cost
2010s	Building satellites fast and cheap enough
2020s	Connectivity — getting the data down
2030s	Coordinating millions of autonomous assets, links, and decisions

For fifty years, the rational way to run a space program was vertical integration: one organization owning the spacecraft, the radios, the ground stations, the operations software, and the scheduling systems. That was the correct architecture — when every fleet was small and every operator controlled nearly every component.

The next generation looks fundamentally different. No one owns all of it anymore. Over the last decade, every layer of the stack unbundled into an ecosystem of its own:

- **Sovereign space programs** are proliferating — nations that want assured access and operational control without building a vertically integrated industrial base from scratch.
- **Commercial operators** fly constellations of hundreds, built from multi-vendor buses, and sell services to governments; **defense missions** ride commercial hardware. Dual-use is the norm, not the exception.
- **Ground stations became services.** Commercial GSaaS networks, national assets, and partner teleports rent by the pass — and **optical ground networks** are arriving with entirely different availability physics.
- **Space situational awareness became a service** — conjunction warnings and catalog data stream in from commercial and government providers, each expecting to reshape today's schedule.
- **Weather, flight dynamics, simulation, and mission planning are becoming software services**, each excellent, each independent, none integrated.

- **Launch became routine**, collapsing the interval between "funded" and "operational" — and moving the bottleneck squarely to operations.
- **Government cloud became real infrastructure** — sovereign regions with federal authorizations, ready to host mission systems that once required dedicated facilities.
- **AI became operationally useful** — able to plan, summarize, and act, if and only if it is grounded in real operational state and bounded by policy.

Each unbundling was rational. Together they created a coordination problem no vertically integrated toolchain was designed for: sovereign governments, commercial operators, infrastructure providers, and specialized software vendors now operate across shared but independently governed systems. The software architecture of space changed before the operational architecture did.

The future of space is unlikely to be dominated by a single vertically integrated ecosystem. The value shifts from owning every component to orchestrating them — securely, reliably, and at global scale.

Every lost gigabit is lost forever

What does operational throughput mean concretely? Unlike terrestrial networks, orbital opportunities cannot be retried. A missed contact window, a degraded optical downlink, a poor routing decision, a weather cell nobody planned around — each destroys transmission opportunities that never come back. The pass is over; the geometry is gone. Every unnecessary handoff, every poor station selection, every avoidable rain fade, every unused minute of visibility is information lost forever.

That is the metric this platform exists to move: **useful gigabits returned to Earth** — mission throughput per unit of infrastructure. The industry has optimized dollars per kilogram, latency, revisit, and spectral efficiency. The 2030s will be won on operational throughput, and it is a coordination problem, not a hardware problem.

> **The bet.** The next decade of space will not be won by the organizations with the best satellites. It will be won by the organizations that can coordinate the most complex operational ecosystems. Hardware is modular; ground, SSA, weather, and simulation are services; mission planning is software. The scarce resource is no longer access to infrastructure — it is the ability to orchestrate it.

03

Mission operations is breaking

The industry's new architecture has an operational consequence: the day-to-day practice of flying satellites is breaking. Not because operators got worse — because the problem changed shape faster than the tools.

3.1 Scale broke the console-per-satellite model

Operational satellite counts have grown by an order of magnitude in a decade, and constellations of hundreds — owned by single operators — are now normal. A human can watch one spacecraft. A team can watch ten. Nobody watches four hundred: at constellation scale, every "rare" anomaly is a weekly event, every pass conflict is a scheduling problem in combinatorial space, and the marginal cost of another operator seat grows faster than the marginal revenue of another satellite. Fleet operations must become software, or fleets stop growing.

3.2 The ground segment fragmented

The ground segment used to be a thing you owned. Now it is a market. Each provider has its own scheduling API, its own conventions, its own failure modes — and today the integration burden lands on the operator. Every operator builds the same brittle adapters to the same providers, and none of that work differentiates their mission.

3.3 Optical changes the physics of availability

RF links degrade gracefully in weather; optical links vanish behind a cloud. As optical downlink becomes the economic answer to sensor data volumes, availability stops being a link-budget margin question and becomes a *routing* question: which site has clear sky in the next twenty minutes, and can the schedule move there in time? That decision must be made continuously, against live weather, for every pass. No human loop closes that fast.

3.4 Everything else got harder at the same time

- **Heterogeneous fleets.** Mixed buses, radios, and payload generations in one constellation — each with different telemetry dialects and constraints — through mergers, iteration, and multi-vendor procurement.
- **TT&C at scale.** Telemetry, tracking, and command across hundreds of assets and dozens of stations is a coordination problem measured in thousands of decisions per day.
- **SSA is a live scheduling input.** Conjunction warnings arrive daily from multiple providers; each one potentially invalidates a maneuver plan, a schedule, and a week of contact bookings downstream — and the replanning is still largely manual.

- **Weather is an operational input.** Rain fade at Ka-band, cloud cover for optical, solar activity for everything — weather is no longer context; it is a scheduling constraint that changes hourly.
- **Multi-operator, multi-government reality.** Coalition operations, hosted payloads, and shared infrastructure mean the trust and isolation boundaries of single-operator tooling simply do not exist.

> **The compounding effect.** Any one of these is manageable. All of them together mean the number of operational decisions per day has grown superlinearly while the tools for making them — consoles, spreadsheets, email, tribal knowledge — have not changed in twenty years. The bottleneck of the space economy is no longer launch. It is operations.

04

The thesis: an operating system for space operations

What broke is not any single tool — it is the absence of a layer. Spacecraft have flight software. Ground stations have modems and schedulers. What is missing is the operating system above both: the layer that owns state, allocates resources, enforces policy, and turns intent into execution.

ConstellationOS is that layer. Like any operating system, it has a definite anatomy:

OS CONCEPT	IN CONSTELLATIONOS
State	The digital twin and the operations graph — every asset, gateway, pass, forecast, command, and telemetry point, connected (Sections 8–9)
Scheduler	Contact scheduling, station selection, and traffic routing across the federated ground network (Sections 14–15)
Drivers	Normalizing adapters for spacecraft telemetry dialects, ground provider APIs, weather, and SSA feeds — written once, by us, not per-operator
Kernel guarantees	Physics: deterministic propagation, link budgets, and capacity models that bound everything above them (Section 11)
Intelligence	Physics-informed ML that forecasts what deterministic models cannot (Section 12)
Shell	The agentic Console, APIs, and AI copilots through which operators express intent (Section 13)
Permissions	Policy envelopes, tenant isolation, and audit — autonomy that is bounded and attributable by construction (Sections 16–17)

> **Mission intent is the abstraction.** Operators should not say "schedule these eleven contacts on these stations." They should say "*download tomorrow's imagery before 0900*" — and the operating system should predict the weather, score the gateways, forecast the links, select the stations, book the contacts, command the passes, monitor execution, reroute around failures, and file the evidence. Intent in; operations out; audit throughout.

There is a precedent for this architecture, and it deserves its own chapter — because it is not a feature of the platform. It is the category.

The rest of this paper is evidence for this thesis: the neutral layer the industry now requires, the platform and its core abstractions — the twin, the operations graph, simulation, physics, ML, AI, federation — the

workflows they enable, and the security architecture that makes all of it deployable by organizations operating critical national infrastructure.

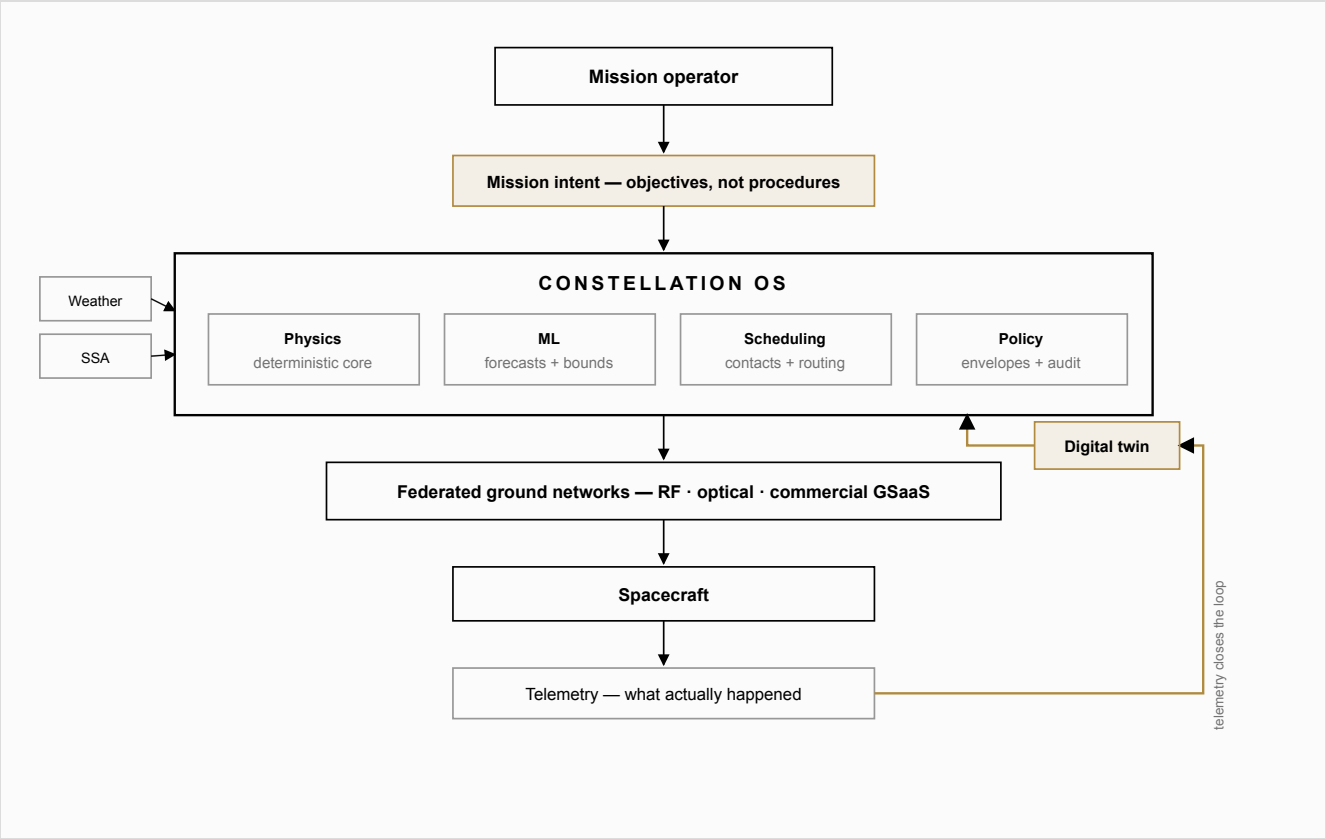

Figure 1. The intent loop: operators express objectives; ConstellationOS — fed continuously by weather and SSA — translates them into scheduled, policy-bound operations across federated infrastructure; telemetry feeds the digital twin, which improves the next decision.

05

The ML-defined network

Constellation is building the first ML-defined, software-defined network for space infrastructure. That sentence is precise, and unpacking it explains why this platform looks the way it does.

5.1 The lineage

Terrestrial networking has already walked this road, one abstraction at a time:

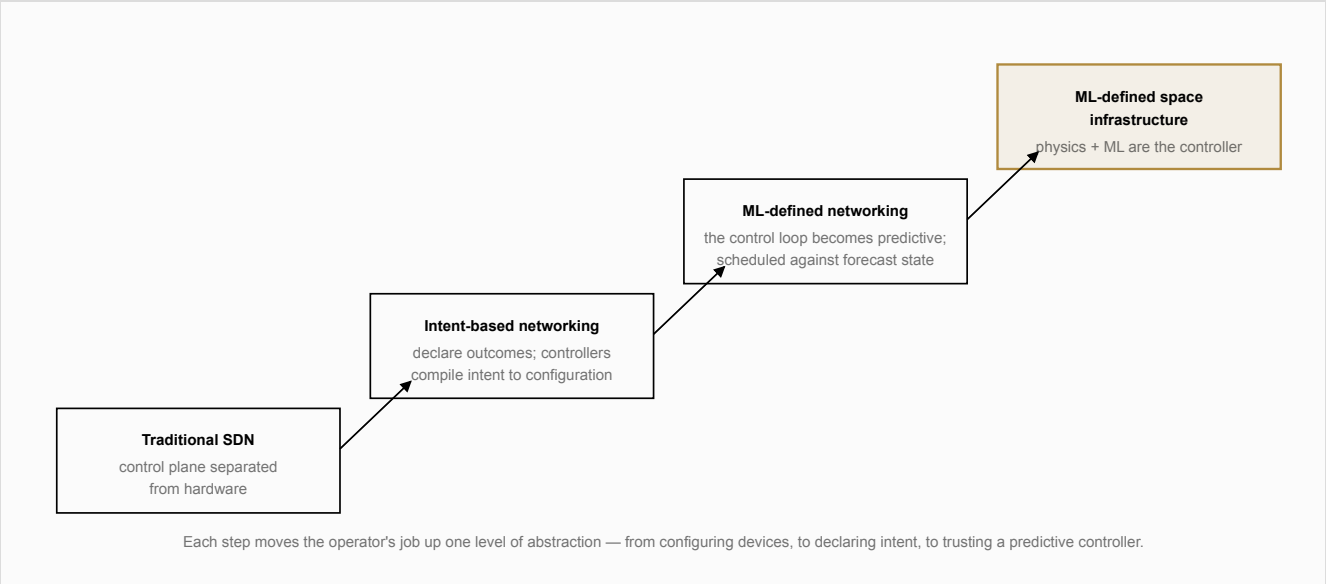

Figure 2. The lineage: SDN separated control from hardware; intent-based networking separated outcomes from configuration; ML-defined networking makes the controller predictive. Space forces the final step.

Traditional SDN separated the control plane from the boxes: routing became a program, not a per-device configuration. **Intent-based networking** raised the interface: operators declare outcomes and a controller compiles them into configuration, continuously verifying the network matches the declaration. Both assume something space does not offer: a network whose topology holds still.

5.2 Why space forces the final step

A space network's links move at 7.6 kilometers per second. Its topology is not configured — it is computed, from orbital mechanics, hours or days ahead. Its link capacities are not fixed — they are functions of elevation, rain, scintillation, and solar activity that change by the minute. In such a network, a reactive controller is always too late: by the time a fault is observed, the geometry that caused it is gone and the window to act with it.

So the controller must be *predictive*. Deterministic physics computes the future topology — where every asset and every possible link will be. Machine learning forecasts what physics cannot: how each specific link will actually perform when it gets there, with calibrated uncertainty. Scheduling, routing, and failover are then solved against the network's *forecast state*, not its observed state — contacts booked on predicted margin, traffic rerouted before the fade arrives, capacity planned on the distribution rather than the average.

Inter-satellite links make this literal. With optical crosslinks, a constellation stops being a set of independent downlink problems and becomes an orbital mesh — nodes moving at orbital velocity, edges appearing and disappearing as line-of-sight and range allow, capacity varying per edge. Data no longer has to come down through the satellite that collected it: it can traverse the mesh and exit wherever the network decides is best — the clear-sky gateway, the uncongested provider, the sovereign site the policy requires. Routing through that mesh, minutes ahead of the geometry, is exactly the problem an ML-defined controller exists to solve — and one no console-and-spreadsheet operation can touch.

That is what "ML-defined" means here: not a network with some ML attached, but a network whose control loop runs on prediction — physics for what is possible, learned models for what is likely, policy for what is allowed. It is the only control architecture that matches the physics of the medium.

5.3 The category

Networking, however, is only one subsystem of what this control plane coordinates: spacecraft, ground stations, optical networks, SSA, TT&C, scheduling, routing, weather, autonomy, policy, governments, and commercial providers. The category this paper describes is **programmable space infrastructure** — assets addressed by intent, coordinated by a predictive controller, governed by policy, and accounted for in an operations graph.

> **The future of space will be coordinated, not commanded.**

06

The neutral operations layer

Constellation is not a ground network, not an SSA catalog, not a weather service, not an analysis tool, and not a flight software stack. It is the layer above them — and its neutrality is not a posture, it is the product working as designed.

5.1 Why neutrality matters

- **Honest optimization requires a neutral seat.** A ground provider optimizing your schedule will find their own antennas remarkably often. Only a layer that owns no infrastructure can select stations purely on predicted margin, cost, and mission intent — across every provider you use.
- **Sovereignty requires it.** Governments will not put mission control inside another nation's — or another operator's — vertically integrated stack. A neutral layer with sovereign deployment postures is the only architecture they can adopt.
- **The ecosystem requires it.** Providers integrate with a neutral layer precisely because it is not their competitor. Neutrality is what lets one control plane speak to every ground network, SSA feed, and weather source at once.

5.2 Composition, not replacement

Enterprise and government customers already have systems they trust: flight software, an SSA provider, mission planning tools, ground contracts, telemetry historians. Constellation does not ask them to replace any of it. It composes them — normalizing each into the operations graph, orchestrating across them, and handing back decisions, schedules, and evidence through the interfaces those systems already speak. Adoption is additive: the platform earns responsibility system by system, never demanding a rip-and-replace to deliver value.

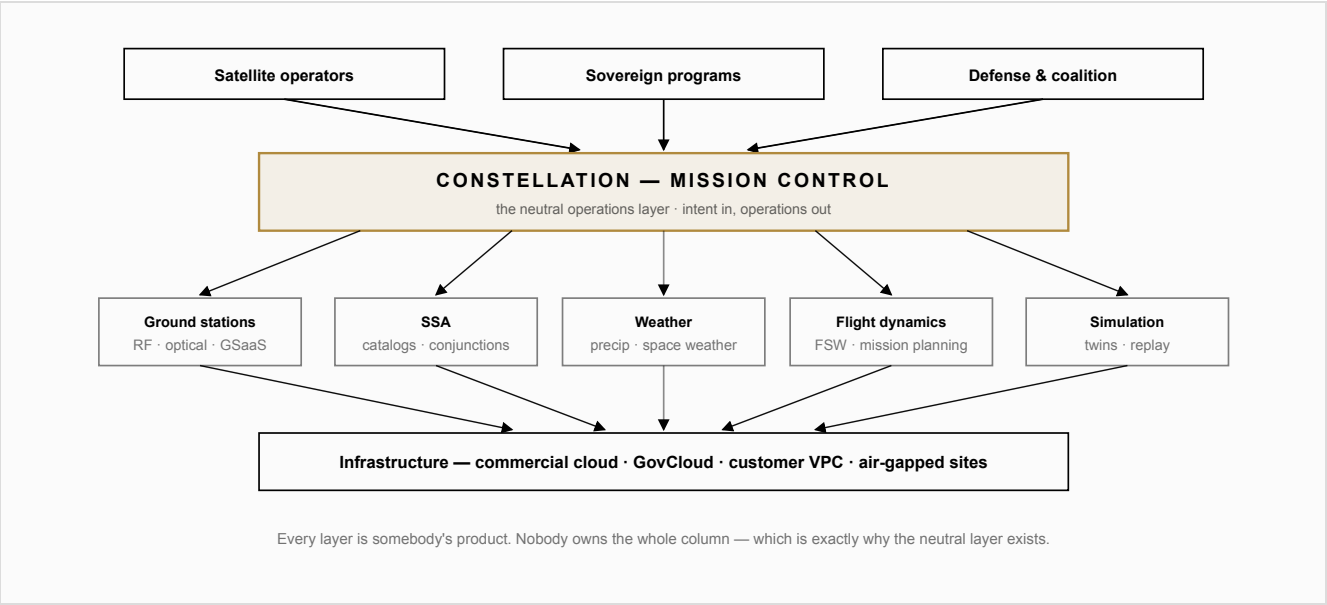

Figure 3. The ecosystem: operators and programs above, composed services below, one neutral mission control layer in between.

07

The Mission Control platform

The platform exposes one contract through four surfaces — an agentic console, enterprise APIs, fully managed mission operations, and per-fleet digital twins. There is no second, internal API: the Console, Mission Operations, and the twins are built on the same contracts customers integrate against. What is demonstrated is what is sold is what is operated.

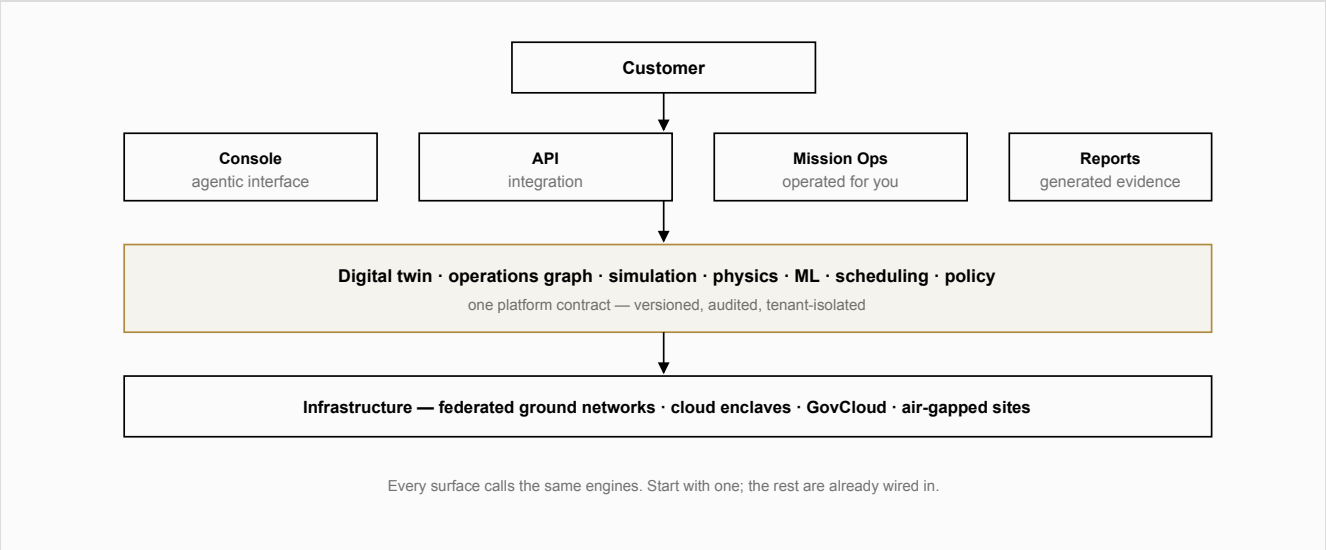

Figure 4. Product surfaces over one platform contract.

Figure 5. The Constellation Console: live fleet topology, link state, and the operator assistant over one 3D operational picture.

08

The digital twin

The twin is not a feature of ConstellationOS. It is the core abstraction everything else reads from and writes to — the operational source of truth.

Every asset, every gateway, every pass, every forecast, every prediction, every command, and every telemetry point exists inside the twin. When the Console renders a fleet, it renders the twin. When the scheduler books a contact, it books it against the twin's predicted geometry and link state. When a forecast is made, it is stored in the twin next to the telemetry that will later prove it right or wrong. There is one model of the world, and everything operates on it.

> **Everything is the twin.** Scheduling reads from it. The AI agent reads from it. Physics writes to it. Simulation *is* it, run forward. Reports query it. Mission planning updates it. SSA enriches it. Telemetry corrects it. If a capability in this paper seems to work smoothly with every other capability, this is why: they are all operating on the same object.

7.1 What the twin holds

LAYER	CONTENTS	SOURCE
Structure	Spacecraft (buses, radios, panels, constraints), ground stations, providers, links	Customer specs + our adapters
State	Orbits, attitudes, link SNR, utilization, power, health	Live telemetry, normalized
Environment	Weather nowcasts, space weather, conjunction geometry	Weather & SSA services
Predictions	Pass geometry, link forecasts (p10/p50/p90), demand, weather impact	Physics + ML (Sections 11–12)
Decisions	Schedules, station selections, routing, maneuvers, commands	Scheduler + operators + agents
History	Every prior state, decision, and outcome — replayable	Immutable event record

7.2 Calibrated against reality, continuously

A twin that drifts from reality is a liability. Live telemetry replays into the twin continuously: observed link quality is compared against the twin's prediction, and the residual — the gap between model and reality — becomes both an anomaly signal (divergence flags problems before threshold alarms fire) and training data (Section 12). As hardware ages, the twin ages with it.

7.3 Decisions are tested in the mirror first

The twin is where operational decisions are validated before they touch the fleet. A maneuver plan, a new contact schedule, a conjunction response — each can be executed against the twin, propagated forward under full physics, and evaluated for consequences. The what-if analysis that used to be a week of engineering becomes a query.

> **Fleet-specific, not generic.** Twins are built from your hardware specifications, your constraints, and your telemetry. The forecast for your Ka-band radio in rain is calibrated on your radio's history — not a datasheet ideal.

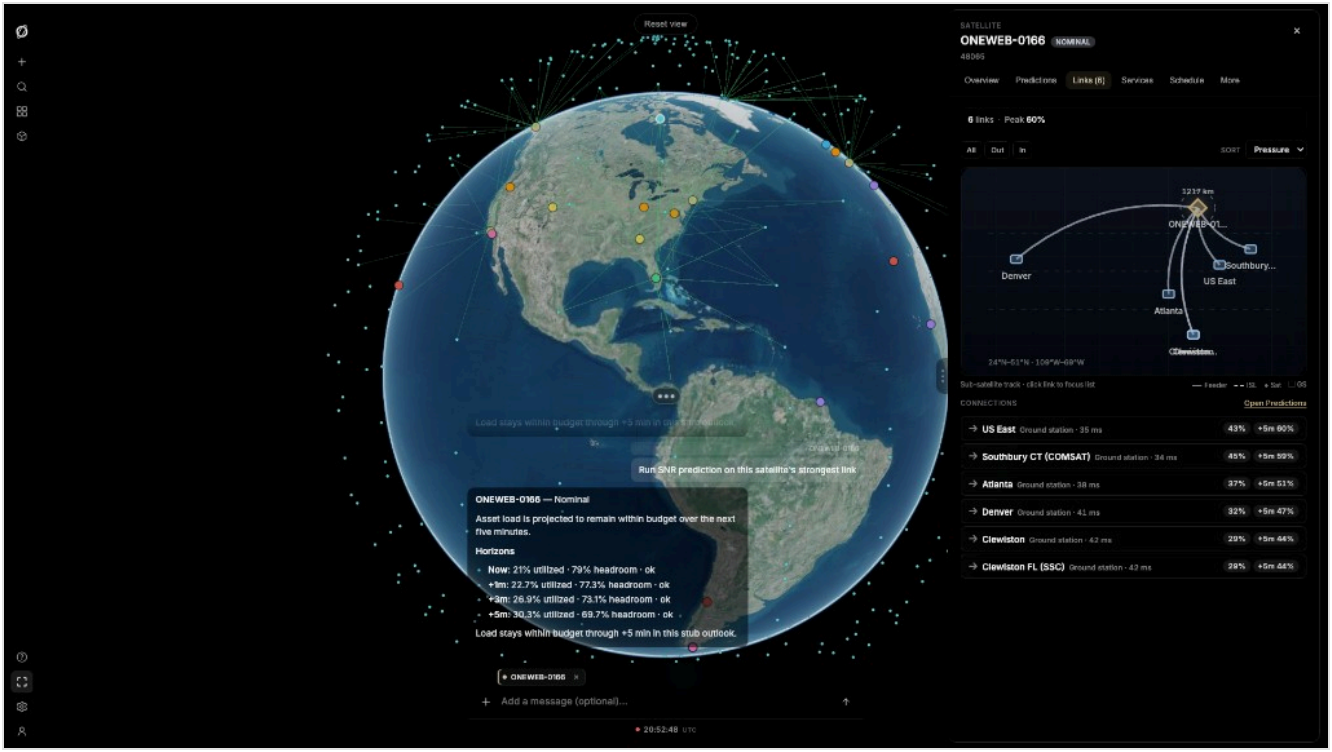

Figure 6. The twin's link layer: live fleet-to-ground topology with per-link state, the substrate for scheduling and routing decisions.

7.4 The twin as audit instrument

Because every forecast, decision, and outcome lands in the twin with provenance, any operational moment can be reconstructed after the fact: what the system believed, what it predicted, what policy allowed, what was done, and what actually happened. For programs that must defend decisions — to regulators, to insurers, to commanders — the twin is the evidence.

09

The operations graph

The twin mirrors the physical world. The operations graph is the larger structure it lives in: everything operational, connected. If the twin is the platform's state, the graph is its ontology — and over time, its deepest moat.

Assets, ground stations, providers, weather cells, conjunction events, policies, predictions, schedules, commands, people, and reports are not rows in separate databases — they are nodes in one graph, with edges that mean something: *this contact was scheduled on that station because of this forecast, under that policy, approved by this envelope, superseding that plan when this conjunction warning arrived.*

9.1 What lives in the graph

NODE CLASS	EXAMPLES	TYPICAL EDGES
Physical	Spacecraft, radios, antennas, ground stations, optical terminals	carries, hosts, links-to, owned-by
Environmental	Weather cells, space-weather events, conjunction geometries	threatens, degrades, invalidates
Predictive	Link forecasts, pass geometry, demand projections	predicts, bounds, supersedes
Decisional	Schedules, station selections, routes, maneuvers, commands	caused-by, approved-under, displaces
Governance	Policies, envelopes, tenants, operators, approvals	authorizes, constrains, attributes
Evidential	Telemetry windows, outcomes, reports, audit records	verifies, contradicts, documents

Governance living *inside* the graph is what makes bounded autonomy real rather than aspirational: a policy is not a PDF — it is a node with edges to every decision it authorized, and tenancy is not a configuration — it is a hard partition of the graph itself.

9.2 The graph is the platform's real interface

Every surface in this paper is a view over the graph. The Console renders it. The APIs query and mutate it. The scheduler solves over it. Reports are frozen traversals of it. The AI agent (Section 13) does not answer from documents; it answers from the graph — which is why "why did we move that contact?" has a literal, inspectable answer instead of a plausible one. The graph is what makes the hard questions cheap:

- **Traceability.** Walk backward from any action to the forecast, policy, and data that produced it — the audit trail is a graph traversal, not an archaeology project.
- **Impact analysis.** Walk forward from any change — a station outage, a conjunction warning, a new policy — to everything it invalidates, before it invalidates it.

- **Cross-domain queries.** "Which passes over Europe next week are exposed to both rain risk and a tracked conjunction?" is one query, because weather, SSA, and schedules share a graph.
- **Counterfactuals.** Fork the graph, apply a hypothetical — a new station, a failed asset, a different policy — and simulate forward (Section 10). What-if analysis is a graph operation.

9.3 Why it compounds

Enterprise software history has one recurring lesson: the durable asset is the connected model of the domain, not any single application on top of it. The application layer gets rebuilt every few years; the ontology underneath it accrues value with every entity, relationship, and outcome it captures.

The operations graph compounds the same way. Every provider integration adds node classes every future customer inherits. Every mission adds edges — decisions linked to forecasts linked to outcomes — that make the next mission's planning sharper. Every anomaly, reroute, and recovery becomes queryable precedent. A new tenant does not start from an empty schema; they start from an ontology already shaped by every operational situation the platform has encountered — while their own data stays entirely their own (Section 17).

Competitors can copy a feature. An ontology that has absorbed years of operational reality across fleets, providers, and regimes is not copyable — it has to be lived. That is the moat.

10

Simulation

Simulation is the twin run forward, backward, and counterfactually. It is not a demo mode — it is how decisions get de-risked and how models get trained.

- **Create constellations.** Author a fleet — real or hypothetical — from orbital elements or design parameters, and propagate it under full physics: SGP4, eclipse, Doppler, link budgets, capacity.
- **Replay history.** Any window of recorded fleet telemetry replays through the engine — the ops-to-sim loop that lets teams study exactly what happened, at any speed, from any angle.
- **Inject failures.** Take down a station, degrade a radio, add a weather front, fail a reaction wheel — and watch the operational consequences propagate through schedules, links, and capacity.
- **Evaluate coverage and capacity.** Revisit, service availability, unserved demand, and beam planning over any region and any architecture.
- **Compare architectures.** Two candidate constellation designs, same demand model, same weather regime — compared on physics, not on vendor slides.
- **Generate synthetic regimes.** Deep fades, storm passages, and solar events occur rarely in real telemetry but dominate operational risk. The engine synthesizes them at ITU-R fidelity for model training (Section 12).
- **Produce reports.** Every simulation can emit a structured report — assumptions, results, and provenance — as a shareable artifact.

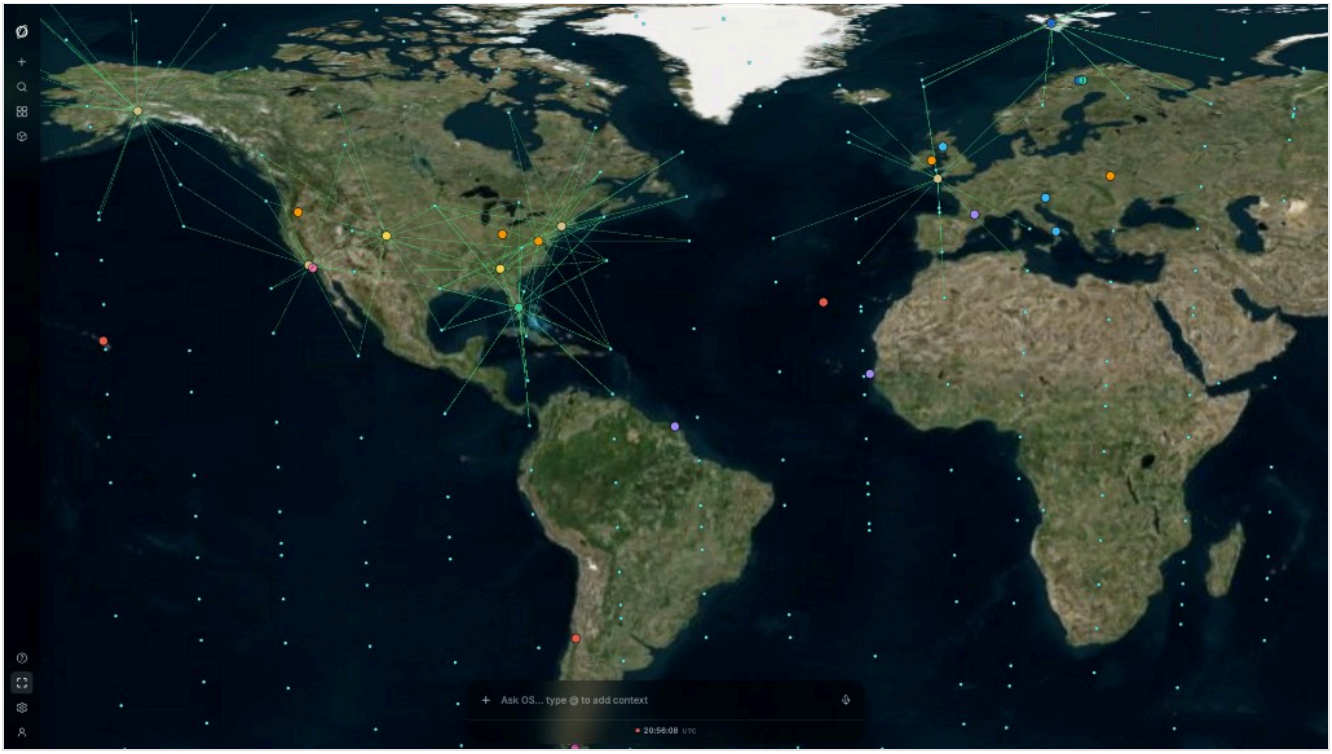

Figure 7. Coverage analysis over a simulated constellation: revisit and service availability rendered from deterministic propagation.

> **Determinism is the point.** Fixed timesteps, seeded randomness, ordered evaluation: the same scenario produces the same result, bit for bit. That is what makes simulations usable as regression tests, as certification evidence, and as ground truth for ML.

11

Everything flows from physics

Every capability in ConstellationOS — every forecast, every schedule, every report — is downstream of a deterministic physics cascade. Nothing in the platform is an opaque score.

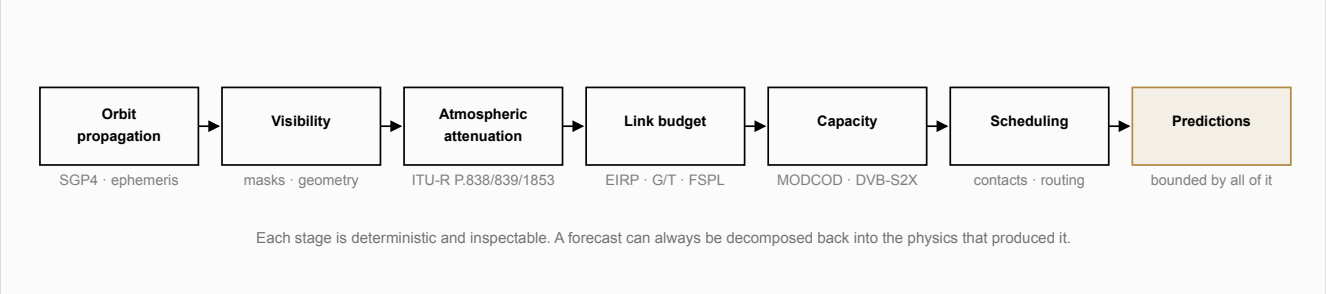

Figure 8. The physics cascade. Every downstream capability inherits its correctness from this chain.

The implementation is uncompromising on the details that matter operationally: SGP4 propagation with proper frame and time handling; slant-path geometry and Doppler; ground-station visibility and inter-satellite line-of-sight gated by geometry and range; rain attenuation via the ITU-R P.838/P.839 chain with P.1853 time-series synthesis for realistic fade dynamics; tropospheric scintillation; sun-noise and eclipse modeling; and full link budgets from EIRP and G/T through free-space path loss to modulation-and-coding capacity. Author in human units, compute in canonical frames, convert at one explicit boundary. One discipline the platform enforces throughout: geometric visibility is a topology primitive, not a link — a crosslink or pass only counts as capacity once the link budget closes.

$$\mathrm{SNR}_{\mathrm{clear\text{-}sky}} = \mathrm{EIRP} + G/T - \mathrm{FSPL}(d, f) + 228.6 - 10 \cdot \log_{10}(B)$$

This clear-sky link budget — computed identically in training and serving — is the anchor for everything the ML layer does, as the next section describes.

12

Physics-informed machine learning

Physics decides what is possible. ML decides what is likely. The architecture makes it impossible to confuse the two.

12.1 Residual learning: bounded by construction

Models are trained on the residual $r(t) = SNR_{observed}(t) - SNR_{clear\text{-}sky}(t)$ — the component physics cannot explain — and forecasts are reconstructed by adding the deterministic term back at each horizon. The learned component is small, inspectable, and structurally incapable of contradicting the link budget: it never sees anything else to learn.

12.2 The feature contract

Every link forecast runs on a versioned contract of physical signals spanning orbital geometry, atmosphere and weather, space weather, hardware configuration, and link history — plus deterministic future covariates: the ephemeris geometry at each horizon, the one thing about the future you are allowed to know. The contract is validated end to end; drift between training and serving fails the build, not the forecast.

12.3 The model portfolio

- **Link-quality forecaster.** Probabilistic sequence models (recurrent trunks with temporal attention, per-horizon heads) emitting full distributions at t+1/3/5 minutes, plus dedicated classifiers for fade onset (≥20%/≥50% drops) and usability thresholds.
- **Demand & capacity forecaster.** Utilization per point of presence with learned diurnal and weekly structure.
- **Station selection.** Scores feasible stations per pass across the federation — the decision model behind intent-level scheduling.
- **Gradient-boosted quantile models.** Residual-dynamics challengers and per-customer calibration.
- **Physics baseline tier.** A deterministic link-budget + orbit-projection forecaster wrapped in conformally calibrated intervals — the production floor learned models must beat, on pre-registered fade regimes, before they ship.

12.4 Calibrated uncertainty

Every forecast ships as p10/p50/p90 with empirical coverage gated into a 70–90% acceptance band, conformally adjusted against live residuals per fade regime, with explicit breach probabilities against operational thresholds. A scheduler does not need the future; it needs an envelope it can trust.

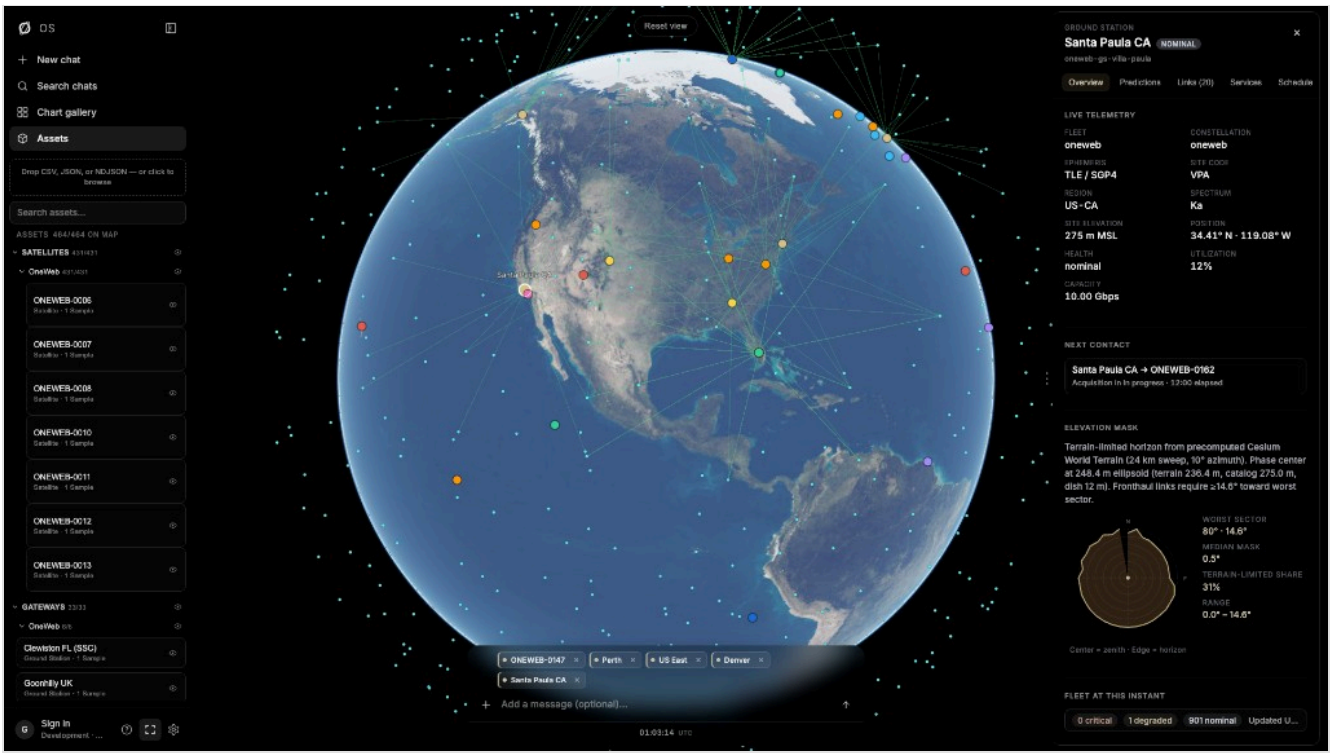

Figure 9. Link forecasts in the Console: median and calibrated interval, decomposable to the physics and features behind them.

12.5 Promoted like flight software

Models are calibrated per site and per fleet, evaluated on held-out operational data against the physics baseline, and promoted through the same gated, reversible release discipline as the rest of the platform — with live scoring against actuals feeding the next candidate. The champion stays until beaten.

Serving is telemetry-grounded and honest: features are built from live tenant telemetry, thin data produces a refusal rather than a guess, and every response carries provenance — model version, data lineage, and the basis for the forecast.

12.6 The learning flywheel

Every operational decision — successful or unsuccessful — becomes another labeled example inside the digital twin: the forecast, the action taken, and the telemetry that followed. Fleet-specific models remain calibrated on each customer's telemetry, inside that customer's enclave. What compounds across the platform is the infrastructure of learning itself: the residual-learning framework, the feature engineering, the simulation regimes, and the evaluation discipline — each made richer by every climate, hardware generation, and operational regime the platform encounters. Physics is shared; residuals are not. Constellation does not become more valuable because it stores more telemetry. It becomes more valuable because every mission expands the operational knowledge encoded in its physics, simulation, and decision engines.

13

AI and agentic operations

The interface to an operating system for space should not be forty dashboards. It should be a conversation with an operator who never sleeps — grounded in the twin, bounded by policy.

The Constellation Console is agentic today: operators ask questions in natural language and the assistant plans and executes against the same engines and contracts as every other surface — running simulations, analyzing constellations, evaluating links, exploring coverage, driving the operational picture.

13.1 What the agent does

- **Mission planning support.** "Simulate 72 hours of contacts for Shell 1" becomes a propagation run, a feasibility analysis, and a conflict-resolved schedule proposal — with the reasoning attached.
- **Operator copilot.** The agent sees live fleet context — the twin — so answers reflect the fleet as it is now, not a stale export. It can drive the console: select assets, load scenarios, focus the timeline.
- **Automatic reports.** Coverage studies, pass summaries, capacity analyses, and post-incident reconstructions generated as structured documents with provenance.
- **Workflow generation.** Recurring analyses become saved, parameterized workflows — authored conversationally, executed deterministically.

13.2 Why grounding matters

A language model with no world model will happily schedule a contact below the horizon. Ours cannot: every tool the agent calls runs through the same physics engines and policy checks as a human operator's actions. The agent proposes; physics disposes; policy constrains; audit records. Agentic operations inherit the platform's guarantees rather than bypassing them.

> **The trajectory.** Today the agent analyzes, simulates, and recommends. As trust accumulates — earned through the same shadow-then-envelope discipline as our other automation — the agent graduates from copilot to executor for well-bounded, reversible operations, always inside operator-defined envelopes, always attributable. Operator → AI → Constellation → infrastructure is the command path we are building toward.

14

The federated ground layer

Every operator today builds the same integrations to the same providers — and none of that work differentiates their mission. Federation replaces per-operator integration with one logical network.

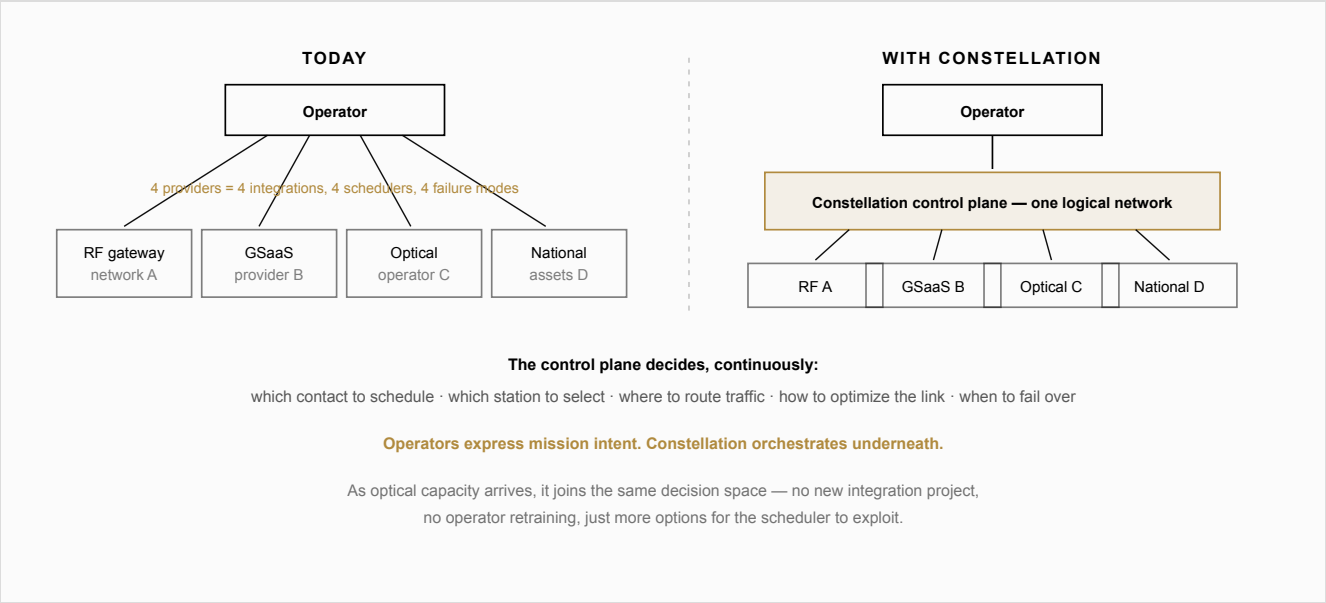

Figure 10. Federation: from per-operator integration burden to one logical ground network under a single control plane.

Federation is more than adapters. The control plane holds a unified model of every station's capabilities, availability, weather exposure, and cost; predicts link performance per candidate site; and treats the entire federation as one resource pool to optimize against mission intent — maximizing uptime, throughput, and resilience across providers a customer never has to think about individually.

And the federation does not stop at the ground. Inter-satellite links extend the same decision space into orbit: with crosslinks in the topology model, the question is no longer "when can this satellite see a station?" but "what is the best path from this data to the ground, across the mesh and out through any gateway in the federation?" Ground selection, ISL routing, and contact scheduling collapse into one optimization — which is precisely why they belong in one control plane.

15

Operational workflows

Architecture is evidence. Workflows are the product. Three scenarios, end to end.

15.1 "Download tomorrow's imagery"

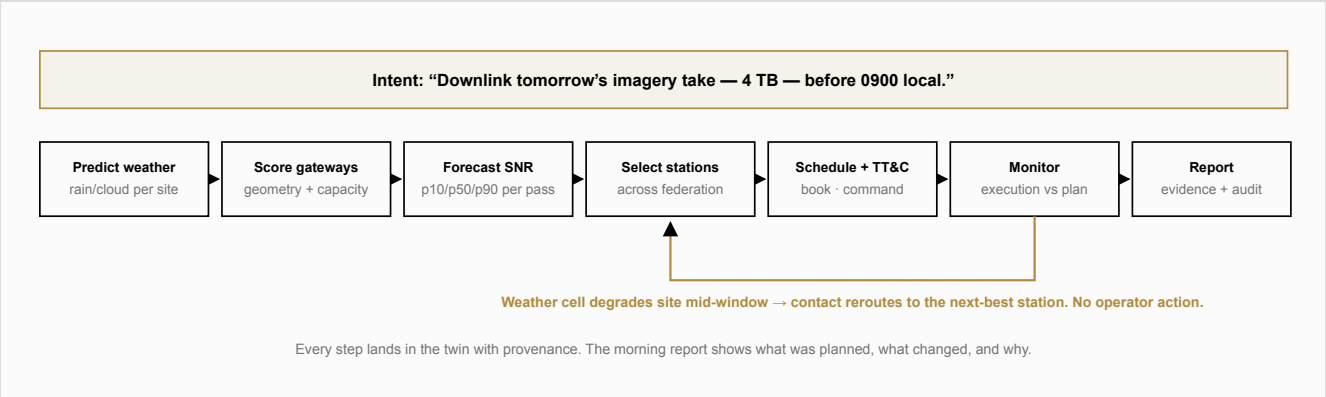

Figure 11. One sentence of intent becomes a monitored, self-healing operational plan.

15.2 Anomaly, contained

02:47 UTC — twin divergence flags wheel momentum trending high on one spacecraft, hours before a threshold alarm would fire. The platform schedules a desaturation window inside the operator's pre-approved envelope, deconflicts the affected contacts, and executes. The on-call engineer reads about it at 08:00 in a report that shows the forecast, the policy that authorized the response, the actions taken, and the telemetry proving recovery. Nothing woke anyone up, and nothing happened outside the envelope.

15.3 Conjunction, replanned

A conjunction warning arrives from an SSA provider: a tracked object passes near one spacecraft in 31 hours. The warning lands in the operations graph, which immediately answers the question no human wants to compile at 23:00 — what does this invalidate? A candidate maneuver is validated against the twin; the platform re-plans the four contacts the maneuver displaces, rebooks them across the federation, adjusts the affected downlink schedule, and stages the command sequence for operator approval — because maneuvers sit outside the automation envelope by policy. The operator reviews one screen: the conjunction geometry, the maneuver, the schedule diff, and the evidence. Approval is one action; everything downstream is already coordinated.

15.4 The capacity question, answered before the meeting

A business team asks: can the fleet absorb a new customer's demand profile in the Pacific? An operator asks the Console. The agent loads the demand model against the twin, simulates a representative week under seasonal weather, identifies the two orbits where unserved demand concentrates, quantifies the gateway capacity that would close the gap, and emits the analysis as a report — assumptions, physics, and confidence intervals included. The meeting starts with the answer.

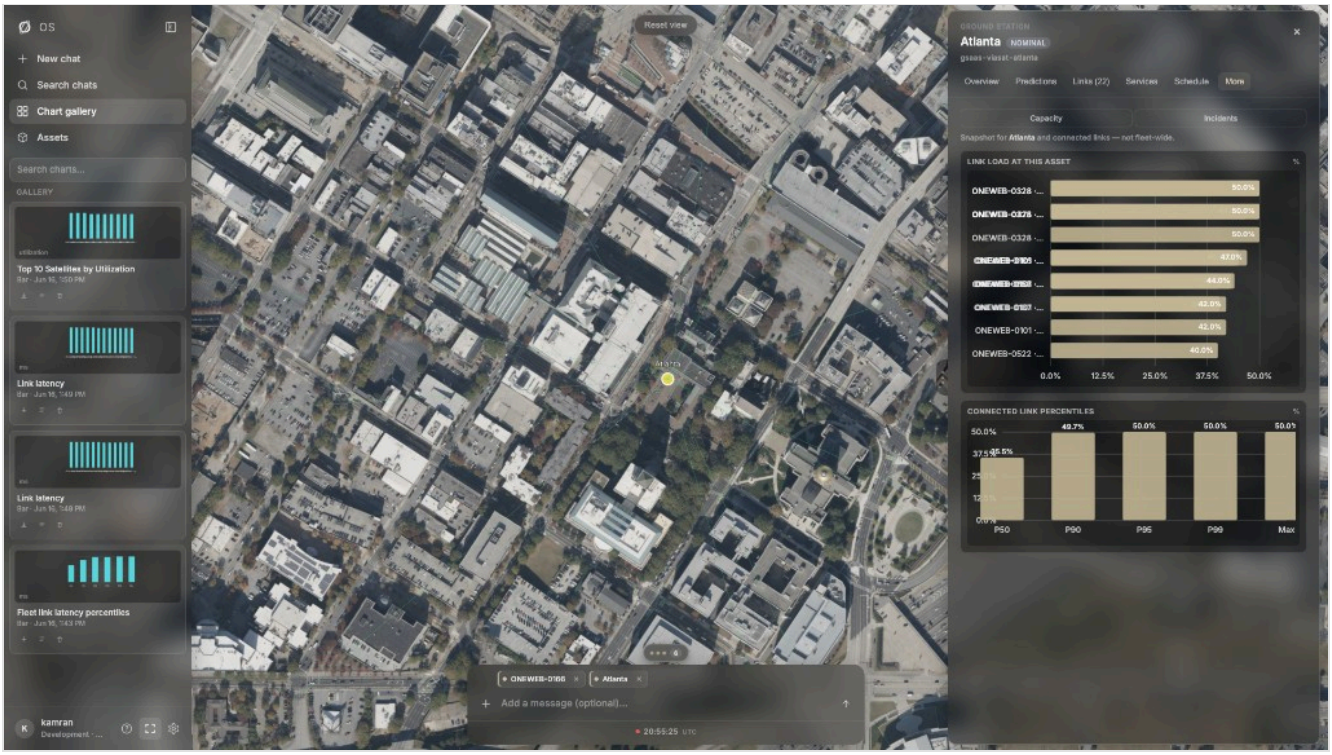

Figure 12. Capacity and utilization analysis in the Console — the substrate for demand questions.

16

Security model

16.1 Identity and access

Human operators authenticate through SSO with MFA. Programmatic access uses per-tenant API keys that are opaque random tokens — verified against server-side memory-hard hashes (argon2), never derivable, never self-mintable, and scoped per capability (for example *telemetry:write*, *predictions:run*). Keys are minted and revoked from the Console; every authenticated call is attributable to a key and tenant.

16.2 Zero Trust posture

No request is trusted by network location — including between internal services. Tenant identity is established at the gateway and re-checked at the data layer, so a request cannot reach another tenant's data even through misconfiguration. Rate limits are enforced per tenant and per source at the gateway.

16.3 Cryptography

TLS protects all data in transit. Data at rest uses envelope encryption under per-tenant customer-managed keys in a cloud KMS built on FIPS 140-3 validated modules. The platform implements no bespoke cryptographic primitives.

16.4 Audit

Every decision surface writes to an immutable, per-tenant audit stream: who (or which policy) acted, on what, when, under which envelope, with which model version. Streams export to customer SIEM and archive systems. The audit trail is designed to reconstruct any operational decision after the fact — including the forecast and provenance that motivated it.

17

Tenant isolation & data boundaries

Isolation is architectural, not configurational. Each tenant's stack — API services, model serving, telemetry stores, model artifacts, keys, and audit streams — is provisioned inside its own virtual private network (and, for dedicated postures, its own cloud account). There are no cross-tenant data paths, no shared encryption keys, and no shared fate: an incident in one tenant's enclave has no propagation path to another's.

BOUNDARY	MECHANISM
Identity	Per-tenant credentials verified at the gateway; tenant pinning re-checked at the data layer
Network	Per-tenant VPC, private subnets, no public ingress to data planes, single audited gateway
Compute	Per-tenant model targets; dedicated serving stacks in dedicated postures
Storage	Per-tenant telemetry stores and artifact buckets, envelope-encrypted under tenant keys
Keys	Customer-managed KMS keys, one hierarchy per tenant
Audit	Per-tenant immutable streams with customer-side export

> **Fail-closed.** Authentication, tenant resolution, and policy checks fail closed. Cross-tenant access is treated as a critical defect class, not a configuration issue.

18

Network & VPC design

The network design is deliberately boring: private by default, explicit where public.

- Data planes (telemetry stores, model serving, artifact storage) live in private subnets with no public ingress.
- A single gateway per environment terminates TLS, authenticates, applies tenant pinning and rate limits, and is the only public surface.
- Egress is explicit and enumerable; security groups are written to be reviewed, not generated.
- Every boundary carries logging and alarms; availability monitoring does not depend on tenant credentials.
- Infrastructure is defined as code and reviewed as code — a new engineer can read the plan and know what exists, who owns it, and how it is monitored.

19

Deployment topologies

One platform contract deploys across five postures. Environments promote through a gated pipeline — development, staging, production — with dedicated tenant stacks deployed as post-production stages of the same pipeline, so every posture runs the same tested artifacts.

POSTURE	INFRASTRUCTURE	RESIDENCY	UPDATES	TYPICAL CUSTOMER
Commercial cloud	Constellation-managed, multi-tenant with full isolation	US commercial regions	Continuous	Commercial operators
Dedicated account	Constellation-managed, single-tenant cloud account	Pinned region	Continuous	Enterprise fleets
Customer VPC	Deployed inside the customer's cloud boundary	Customer's region	Gated releases	Regulated enterprises
Sovereign cloud	AWS GovCloud (US) — operating today with dev/stage/prod environments	Jurisdiction-pinned	Gated releases	Government & defense
Air-gapped / on-prem	Customer-controlled sites, offline artifact bundles	On-site	Offline bundles	Classified-adjacent programs

GovCloud today. The sovereign posture is not a roadmap item: ConstellationOS runs in AWS GovCloud (US) with dedicated development, staging, and production environments mirroring the commercial pipeline.

20

Standards alignment

The architecture and engineering process are designed against the frameworks our customers are assessed under. We do not claim formal certification unless stated explicitly; we design so that certification is a milestone, not a rewrite.

FRAMEWORK	DOMAIN	HOW CONSTELLATIONOS MAPS
ECSS	Space software engineering	Requirements traceability; deterministic simulation as reproducible verification evidence; documented lifecycle.
DO-178C / ED-12C	Safety-critical software principles	Requirements-based testing on asset-commanding paths; severity-graded review gates; envelope-bounded autonomy with explicit criticality.
ARP4754A / ARP4761	Systems & safety processes	Failure-mode analysis per automated function; degradation designed toward operator control; hazard analysis in feature definition.
EN 9100 / AS9100	Aerospace quality management	Configuration-managed traceable builds; gated promotion; nonconformance tracking; supplier inventory.
NIST CSF / 800-53	Security controls	Control-mapped architecture: least-privilege IAM, audit and accountability, integrity monitoring, continuous alarms.
Zero Trust (800-207)	Architecture principle	Per-request tenant authentication; no network-location trust; identity re-checked at the data layer.
FIPS 140-3	Cryptography	Validated cryptographic modules via cloud KMS/TLS; per-tenant customer-managed keys; no bespoke crypto.
FedRAMP	Federal cloud authorization	GovCloud deployment on FedRAMP High infrastructure; small, documentable platform control surface for inheritance.
ITAR / EAR	Export control	Jurisdiction-pinned enclaves; US-person access controls where required; export review in contracting.
CMMC / NIST 800-171	Defense industrial base	800-171 control families implemented in isolation, audit, and encryption design; readiness mapping per family.

21

Operational assurance

The platform is engineered failure-first: for every capability we identify the most likely failure, the highest-cost failure, and the least observable failure, and design so each is easy to detect, diagnose, and recover

from.

- **Determinism.** Fixed timesteps, seeded randomness, ordered evaluation — bit-reproducible runs make regression testing and certification evidence possible.
- **Refusal over guessing.** Thin telemetry fails coverage gates and returns a refusal; models never silently extrapolate past their evidence.
- **Envelopes.** Autonomy operates inside operator-defined envelopes; anything outside escalates to humans with full context.
- **Provenance.** Every forecast and every action carries the model version, data lineage, and policy that produced it.
- **Graceful degradation.** The physics baseline remains available when learned models are withheld — the system degrades toward deterministic behavior, never toward silence.

22

Where space infrastructure is going

The next decade of space infrastructure will look fundamentally different from the last — and every version of that future needs the same thing.

- **Optical links** will replace many RF links, trading weather sensitivity for bandwidth — and moving availability decisions to machine timescales.
- **Spacecraft will increasingly talk to each other**, turning constellations into networks whose topology changes by the minute.
- **Compute will move closer to the edge** — in orbit, on launch vehicles, and across distributed infrastructure — reducing latency and enabling autonomy that cannot wait for a ground loop.
- **Commercial, sovereign, and defense assets will share infrastructure without sharing control**, multiplying the trust boundaries every operation must respect.
- **The number of operational decisions will grow by orders of magnitude** — far past what any operations floor can staff.

These trends have one thing in common: each dramatically increases the complexity of mission operations. Every new spacecraft, optical terminal, inter-satellite link, compute node, SSA provider, and ground station is another resource that must be coordinated in real time. Coordination — not hardware — becomes the limiting factor.

The compute trend deserves particular attention. As compute moves into orbit and constellations evolve from communications systems into distributed compute and sensing platforms, mission operations become increasingly analogous to cloud orchestration. The challenge is no longer operating individual spacecraft — it is scheduling, routing, and coordinating a globally distributed computational infrastructure under the constraints of orbital mechanics. Whoever builds that future will need a control plane that already thinks this way.

> **The claim.** Whether the future contains orbital data centers, autonomous spacecraft, optical relay networks, or fully federated sovereign infrastructure, every one of those futures depends on an intelligent operational control plane. We believe the next generation of space infrastructure will be measured less by how much hardware is deployed and more by how effectively that hardware is coordinated. Constellation is building the ML-defined, software-defined network that coordinates it.

The right analogy is not a bigger mission control center. It is the transition from physical servers to cloud computing. Before the cloud, teams managed machines — racked them, patched them, watched them — one by one. After, they declared what they needed and the control plane arranged it. Space operations sits at the same threshold: before, operators manage satellites, ground stations, links, and contacts

individually; after, they manage intent. Constellation abstracts satellites, ground stations, optical terminals, SSA, weather, and TT&C into programmable operational infrastructure — an orbital cloud, addressed by intent rather than by console.

23

Direction

Not a feature list — a direction of travel. Everything below extends machinery described in this paper; nothing requires a new thesis.

- **Mission autonomy.** Widening envelopes: from automated responses to automated campaigns, earned through the shadow-then-envelope discipline that governs all our automation.
- **Agentic operations.** The copilot graduates to executor for bounded, reversible operations — natural language as the primary command surface, audit as the primary record.
- **Optical networks.** First-class optical scheduling: cloud-aware site selection and preemptive rerouting as optical ground segments scale.
- **Deeper SSA integration.** Conjunction data already enriches the twin and drives replanning (Section 15.3); next, maneuver options are generated and scored automatically, with coordination across operators sharing the same conjunction.
- **Federated scheduling.** From per-operator optimization to cross-federation markets: capacity discovered, priced, and allocated across providers.
- **Autonomous TT&C.** Routine command and telemetry sessions planned, executed, and verified without hands on keyboards — with the audit trail to prove it.
- **Digital certification.** Simulation evidence packaged to certification-grade standards, so the twin becomes part of the compliance story, not just the operations story.
- **Deeper government & enterprise postures.** Expanding sovereign, customer-VPC, and air-gapped deployments as programs demand them.

IN CLOSING

Computing evolved from machine code to operating systems. Networking evolved from leased circuits to software-defined networks. Space operations are beginning the same transition — from procedures executed by hand to intent executed by software.

By 2035: millions of spacecraft. Tens of thousands of ground terminals. Commercial, sovereign, and defense infrastructure operating simultaneously. AI making millions of operational decisions every hour. Humans defining objectives — not procedures. Mission operations becoming software.

We believe the future of space will not be built around individual spacecraft. It will be built around intelligent coordination.

The future of space will be coordinated, not commanded.

CONSTELLATION SPACE CORP · SEATTLE, WA ·
CONTACT@CONSTELLATION.SPACE · CONSTELLATION.SPACE