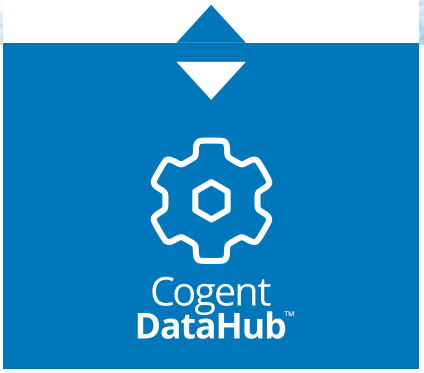
WHITE PAPER

Stop Unifying Everything



A Unified Namespace (UNS) connects all data sources and users across a plant or enterprise, typically with a single broker. The concept is fine, but the design is flawed. When every node trusts every other node, you haven't built resilience, you've built a blast radius. A fractal UNS can replace the monolithic single-broker model to provide self-similar namespaces at every operational level, improving resilience, latency, and security.

Key takeaways

1. The push to connect everything into a single UNS trades operational resilience for architectural convenience. There's a better way.

2. A single centralized broker creates a single point of failure, adds latency, and complicates OT network segmentation.

3. A fractal UNS mirrors the operations hierarchy. Each level — machine, line, site, enterprise, cloud — handles its own data and operates independently during outages.

4. Common objections to a fractal UNS, such as complexity, cost, and apparently multiple sources of truth, fade away when you understand how it works and know the benefits.

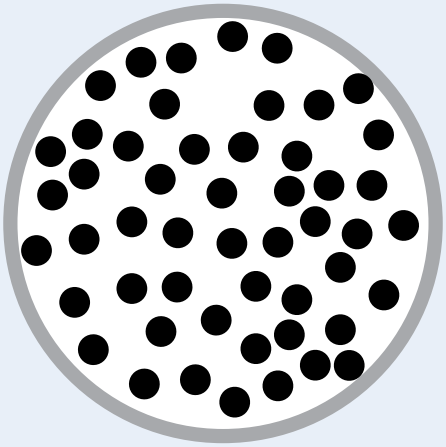
The Promise and the Problem

For the past few years, the industrial automation world has been chasing a single idea: coalesce everything into one namespace. Connect every PLC, sensor, historian, and ERP system into a single, unified data fabric. One namespace to query. One broker to handle protocol conversion, naming conventions, engineering units, security, and edge-to-cloud bridging. It's an ambitious idea — and it solves a real problem. Industrial operations have spent decades building incompatible islands of data, and the cost in engineering hours, delayed decisions, and missed optimization is enormous.

But solving a data accessibility problem by removing all architectural boundaries is like solving a traffic congestion problem by removing all curbs, sidewalks, guardrails and traffic signals. Yes, everyone can get anywhere faster. That's not necessarily good.

When every node in your namespace can reach every other node, you have not built a resilient system. You have given a compromised HMI in Building 4 a direct path to your corporate historian. A misconfigured MQTT client can now flood your broker and take down visibility across the entire plant.

Monolithic
UNS



In traditional OT architecture segmentation was a feature — sometimes an accidental one, but a feature nonetheless. Air gaps and protocol barriers that drove integration teams crazy also meant that a failure or breach stayed local. The standard monolithic UNS design that connects all data sources and users through a single centralized broker trades that containment for convenience, but at what cost? It introduces a single point of failure, increases latency among participants, and makes it difficult to segment OT networks.

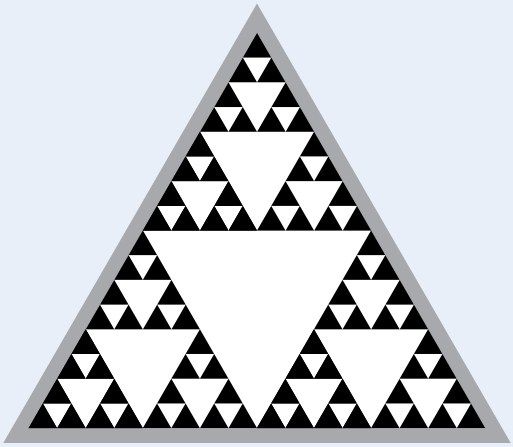
That is not a trade most architects should be making blindly. The architects and engineers who have to live with these systems long-term are asking questions the evangelists often skip:

- What happens when the broker goes down?
- What happens when a network connection goes down?
- What is the blast radius of a compromised node?
- How do you enforce data sovereignty across sites in different regulatory jurisdictions?
- How do you maintain performance when 150,000 tags flow through a single namespace?
- Who owns the namespace schema, and how do you govern changes without breaking downstream consumers?

The Fractal Alternative

There is another way. Your operation is segmented. Machines in a production line are each responsible for some data. Your production line SCADA is responsible for data aggregated from the machines. Factory operations aggregate data from the production lines. Site systems aggregate data from the factories, and enterprise systems aggregate data from multiple sites.

Fractal
UNS



The system is fractal. Each layer is self-similar. It looks and behaves like the others, just at a different scale. Each cell, line, or site maintains its own local namespace, fully functional and autonomous. Data flows upward, downward, or horizontally through defined, controlled pathways — not through open mesh connectivity. You can think of it as similar to the Internet's DNS system: local resolvers handle nearby queries, regional servers handle broader ones, and root servers handle global lookups.

A fractal UNS gives you everything a flat namespace can deliver, and adds the structural resilience that flat architectures inherently lack. The result is an architecture with meaningful properties:

- **Fault isolation.** A broker failure at the plant level does not stop lines from operating. It just reduces visibility at and above that level. A network disruption between sites does not take down local operations. Each layer operates independently and the system degrades gracefully.
- **Contained blast radius.** A compromised node can only reach what it is architecturally permitted to reach. Lateral movement — the defining characteristic of modern ransomware campaigns — is constrained by design, not by policy alone.
- **Defined data sovereignty.** When namespaces are bounded, you can enforce what data crosses which boundaries. This matters enormously for multi-site operations, joint ventures, vendor access, and differing regulatory regimes.
- **Scalable governance.** Schema ownership is distributed. The team running the stamping line owns their namespace. Corporate data architects own the enterprise aggregation layer.
- **Predictable performance.** Traffic stays local until it needs to go broader. High-frequency process data does not compete for bandwidth with enterprise reporting queries.

None of this requires abandoning the UNS vision. Instead, it requires implementing it with the same rigor you would apply to any critical OT system.

Criticisms of a Fractal UNS

Those who argue against a fractal UNS focus on Simplicity, Cost, and the "Single Source of Truth", the primary selling points of a flat architecture.

Some ask about the "Complexity Tax" of fractal architectures. While a fractal UNS provides resilience, it introduces complexity costs in engineering, configuration, and maintenance they say. However, a monolithic UNS is only "simple" until a single misconfigured MQTT client or a compromised node floods the broker. At that point, the "simple" architecture causes a total loss of visibility across the entire plant, at significant cost. In reality, the design of a fractal UNS falls naturally from the system architecture and the data visibility requirements at each level.

Others raise the question of a "Single Source of Truth." In reality, no UNS is a single source of truth. The real source of truth is the physical system, and every communication level above that is a delegate. A PLC isn't truth if a wire is broken; an MQTT broker isn't truth if a network connection is lost or congested. Each level must reliably report loss of authority, such as a broken connection in a lower level, to the levels above.

At best, a UNS is a "single point of contact" for truth at that level. To be reliable, it must confirm that its values are current and correct, or flag them as disconnected. This isn't about monolithic vs. fractal, it's whether the broker and protocol can guarantee data consistency from source to subscriber. MQTT cannot. You need purpose-built software to make any UNS trustworthy.

And what about the Rigidity trap? The claim is that solving security through "architecture" could create a costly rigid system that cannot adapt to the fast-moving needs of a modern digital business. In reality, complexity is not actually increased in a fractal system. Instead, it is distributed to the teams who actually understand the data, such as the engineers running a specific stamping line. Changes can be made on the spot by those responsible for the process.

The following table compares the structural and operational capabilities of a Monolithic UNS against a Fractal UNS architecture point by point.

Capability	Monolithic UNS	Fractal UNS
Core Architecture	**Single, centralized broker** connecting all sources and users.	**Self-similar namespaces** replicated at every operational level (Machine, Line, Site, Cloud).
Failure Mode	**Single Point of Failure:** Broker downtime cascades across the entire plant or enterprise.	**Fault Isolation:** Local layers operate independently; failures do not cascade to other levels.
Security Model	**Open Mesh:** Every node can potentially reach every other node, increasing the "blast radius".	**Contained Blast Radius:** Lateral movement is constrained by architectural boundaries at each layer.
Data Governance	**Centralized:** One team or person must own the entire global schema.	**Distributed:** Schema ownership is delegated to the teams closest to the data (e.g., stamping line team owns their layer).
Performance	**High Latency/Congestion:** High data volume flowing through one broker creates bottlenecks.	**Edge-Optimized:** Traffic stays local until needed, preventing high-frequency data from choking bandwidth.
Data Sovereignty	**Difficult to enforce** across different sites or regulatory jurisdictions in a flat pool.	**Defined Boundaries:** Bounded namespaces allow strict enforcement of what data crosses specific borders.
Data Quality	**Standard MQTT brokers** introduce QoS reliability issues.	**Purpose-built software** can guarantee data consistency.

Design and Security Considerations

The advantages of a fractal UNS come through specific design choices. The software is fundamentally different from a monolithic UNS. You cannot simply connect two or more UNS brokers together and expect the same results. Most UNS brokers use MQTT, whose pub/sub architecture offers a simple, secure way for industrial devices to connect. However, MQTT QoS guarantees do not propagate reliably across more than one broker connection, resulting in a fragile system. This is explained in more detail in our white paper Which Quality of Service (QoS) is Right for IIoT?

Software designed to support a fractal UNS guarantees consistency of data from each source to each user. Data moves seamlessly between the various levels of hierarchy, as each level delegates authority of the data to the level above. If communication between levels breaks at any time, or if one UNS server anywhere in the system goes down, that data is flagged as not connected, and the other UNS servers continue functioning on that basis.

Security teams are increasingly involved in OT architecture decisions, and for good reason. The convergence of IT and OT has extended enterprise threat surfaces deep into plant environments. What was once physically isolated is now networked. A fractal architecture, where each layer enforces its own access controls, addresses this reality structurally rather than relying on policy alone.

If you are designing a UNS implementation today, the questions worth asking are not "how do we get everything connected" but rather:

- **What is the right boundary for each namespace segment?**
- **What data genuinely needs to cross each boundary, and in which direction?**
- **What is the failure mode when any single component goes down?**
- **What is the containment boundary if a node is compromised?**
- **Who governs each layer of the hierarchy?**

A flat UNS can answer some of these questions with policy and configuration. A fractal UNS answers them with architecture, which means the answers hold even when your policies are misconfigured, your documentation is outdated, or your team is responding to an incident at 2 AM.

The Bottom Line

The UNS concept has proven value. Unifying your namespace is the right goal, but building a single, flat, fully-connected namespace to achieve it is the wrong architecture for any operation that cares about resilience, security, or long-term maintainability.

The industry has been so focused on solving the integration problem that it has ignored the failure mode problem. As a system architect or engineer, that is exactly the trade-off you are paid to see clearly.

The push for a Monolithic UNS often prioritizes architectural convenience and "flattening" data silos. However, this trades away operational resilience. In contrast, a Fractal UNS provides the same unified data benefits while restoring the structural segmentation that prevents local issues from becoming enterprise-wide disasters. Additionally, a fractal UNS provides autonomy, and security by replicating a consistent structure at every operational level. Achieving this requires purpose-built software, not simply connecting brokers.

Stop unifying everything.
Start segmenting with intent.

DataHub software from Skkynet is built on a fractal-ready connectivity architecture designed for industrial environments where security and resilience are not optional. Learn more at skkynet.com.

About Skkynet

Skkynet is a global leader in real-time software and services that allow companies to securely acquire, monitor, control, visualize, network and consolidate live process data in-plant or in the cloud. DataHub™, and DataHub™ for Azure, enable secure, real-time data connectivity for industrial automation, Industrial IoT, and Industrie 4.0. Visit **skkynet.com** for more about the company and **cogentdatahub.com** for more about Cogent DataHub.

Skkynet™, DataHub™, Cogent DataHub™, the Skkynet and DataHub logos are either registered trademarks or trademarks used under license by the Skkynet group of companies ("Skkynet") in the USA and elsewhere. All other trademarks, service marks, trade names, product names and logos are the property of their respective owners.