FIREBALL INDUSTRIES

IGNITE YOUR FACTORY EFFICIENCY

WHITE PAPER

Publish Once, Use Everywhere

Unified Namespace architecture for manufacturers: MQTT, Sparkplug, and ISA-95, and how to build one that outlasts the next system you buy.

By **Patrick Ryan** | Fireball Industries | November 2026



EXECUTIVE SUMMARY

Every new system should not mean another integration project.

Most plants move data the way they always have: one connection at a time. The MES reads from the PLCs. The historian reads from the same PLCs. The ERP gets a nightly export, the maintenance system gets a CSV, and the quality database gets a custom script someone wrote in 2017 and nobody wants to touch.

Every one of those connections is a small project, a small contract, and a small point of failure. Each works on the day it is commissioned. Each one breaks quietly when a tag is renamed, a PLC is replaced, or a vendor ships an update. The plant ends up paying an **integration tax** on every improvement it tries to make, and the tax rises with every system added.

A **Unified Namespace** (UNS) replaces that web with a single hub. Every system publishes what it knows, once, to a shared, organized, real-time namespace. Every system that needs data subscribes to it. A new dashboard, a new MES module, or a new AI model connects to one place instead of to every machine, and it sees the plant as it is right now, not as it was at last night's export.

The idea is not new and the parts are not exotic. The UNS runs on open, standardized building blocks: **MQTT**, an OASIS standard and ISO/IEC 20922; **Sparkplug 3.0**, published as ISO/IEC 20237:2023; and the **ISA-95** equipment hierarchy from IEC 62264, which gives the namespace a structure every plant engineer already recognizes.[1,3,5,7] What makes a UNS succeed or fail is not the technology. It is the namespace design, the security around the broker, and the discipline to keep it clean as the plant grows.

This paper explains what a Unified Namespace is and is not, the standards underneath it, how to design the namespace itself, how to secure it, and how to roll it out one line at a time without stopping production.

45

Possible point-to-point links between 10 systems. A Unified Namespace needs 10 connections.

1

Place every system publishes to and subscribes from, organized the way the plant is organized.

0

Inbound connections into the control network when edge nodes publish outbound to the broker.

SECTION I

The Integration Tax

Picture a mid-size plant with a modest software footprint: PLCs on each line, a SCADA or HMI layer, a historian, an MES, an ERP, a maintenance system, a quality system, and a couple of reporting tools. That is ten systems before anyone mentions analytics, AI, or a second site.

In a point-to-point world, any two of them that need to share data get their own link. Ten systems can need up to **45 links**, because the number of possible pairs is n(n−1)/2. Twenty systems can need up to **190**. Not every pair talks, but the pattern holds: the integration work grows with the square of the system count, while the value of each new system grows in a straight line. Somewhere along that curve, a plant stops adding tools, not because the tools are bad but because connecting them costs more than they return.

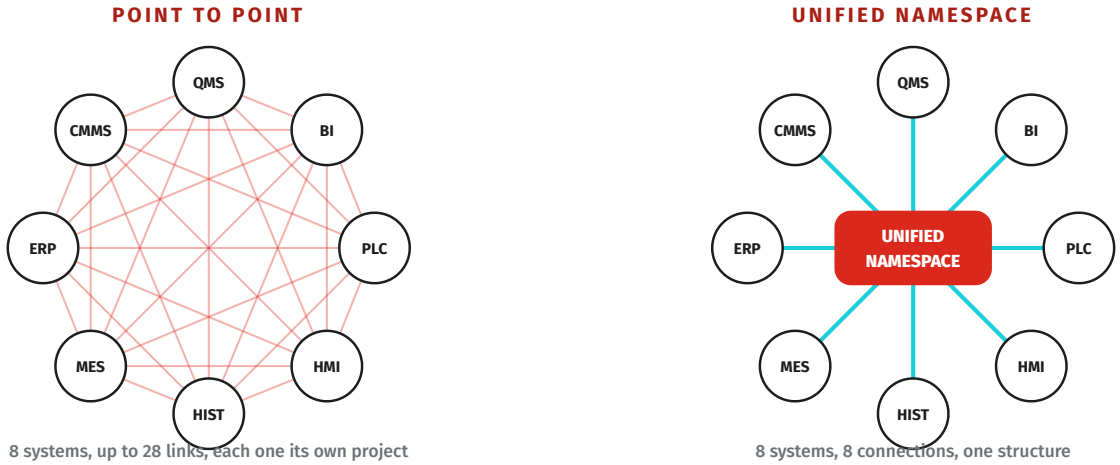

Figure 1. The same eight systems wired point to point and through a Unified Namespace. The left side is drawn with every possible link; real plants have fewer, but every one they do have is maintained by hand.

The tax is paid in more than money

The cost of point-to-point integration does not show up as one line item. It shows up as:

- **Brittle links.** A renamed tag, a replaced PLC, or a firmware update breaks every connection that depended on the old address, and the plant usually finds out when a report comes back empty.
- **Stale data.** Batch exports and polling schedules mean the ERP, the MES, and the floor each see a different version of the truth, separated by minutes or hours.
- **No context.** A tag called `N7:42` means something to the engineer who programmed it. It means nothing to a business system, so every consumer has to rebuild the context on its own.
- **Load on the controllers.** Every system that polls a PLC directly adds traffic to the device the plant can least afford to slow down.
- **Lock-in.** When a vendor owns the connection between two systems, replacing either one means paying for the connection again.

None of these problems is solved by a better point-to-point link. They are solved by changing the shape of the architecture.

SECTION II

What a Unified Namespace Is, and What It Is Not

A Unified Namespace is a single, organized, real-time view of the business, built on a publish and subscribe model. It has four properties that matter:

1. **One place.** Every producer of data publishes to the same namespace, and every consumer subscribes to it. Systems stop talking to each other directly and talk to the namespace instead.
2. **Organized like the plant.** The namespace is structured by enterprise, site, area, line, and cell, so anyone who knows the plant can find the data without a tag list.
3. **Current state, by event.** Values are published when they change, not when someone asks. The namespace always holds the latest known state of everything in it.
4. **Producers do not know their consumers.** A PLC gateway publishes a value once. Whether one system or fifty use it is not the gateway's problem, and adding a fifty-first changes nothing upstream.

At the center sits a **broker**, the server that accepts published messages and delivers them to subscribers. Around the edge sit **edge nodes**: gateways or software next to the equipment that read the PLCs, drives, and instruments over the protocols they already speak and publish what they read into the namespace.

What it is not

Not a historian

The namespace holds current state. History is a consumer: a historian subscribes to the namespace and stores what it needs. The two work together, and neither replaces the other.

Not a data lake

A UNS is not a place to dump every register a PLC has. It carries the data the business uses, with context, and it stays organized because someone owns its structure.

Not the control network

PLCs keep controlling the process. Interlocks, safety functions, and closed loops stay in the controllers and on the control network. The UNS observes the process and informs decisions about it.

Not a product

A UNS is an architecture. It can be built from several brokers, gateways, and platforms, and that openness is the point: no single vendor owns the hub.

> **A simple test.** If adding a new consumer of plant data requires changing a PLC program, an HMI, or an existing integration, you do not have a Unified Namespace yet. If it only requires a new subscription and a permission, you do.

SECTION III

The Standards Underneath

A Unified Namespace is only as durable as the standards it is built on. The good news is that the core pieces are open, published, and mature.

MQTT: the transport

MQTT is a lightweight publish and subscribe messaging protocol. It was created in 1999 by Andy Stanford-Clark of IBM and Arlen Nipper to monitor oil pipelines over satellite links, where bandwidth was expensive and connections could not be trusted to stay up.[4] Those constraints are exactly the ones a plant floor has: many small devices, unreliable networks, and no tolerance for a protocol that floods the wire.

MQTT 3.1.1 became an OASIS Standard in October 2014 and was approved as international standard ISO/IEC 20922 in 2016. MQTT 5.0 became an OASIS Standard in March 2019, adding features such as reason codes, shared subscriptions, message expiry, and user properties.[1,2,3] Both versions remain in wide use. Three MQTT features matter most for a UNS:

- **Topics**, hierarchical names separated by slashes, which let the namespace mirror the plant hierarchy and let consumers subscribe to whole branches with wildcards.
- **Retained messages**, which let the broker hold the last value on a topic, so a new subscriber gets current state immediately instead of waiting for the next change.
- **The last will**, a message the broker publishes on a client's behalf if that client disappears without saying goodbye, so the namespace knows when a source has gone dark.

Sparkplug: state and structure for industrial data

Plain MQTT moves bytes. It says nothing about what is inside them. Sparkplug, an Eclipse Foundation specification, adds what industrial systems need on top of MQTT: a defined topic structure, a defined payload, and a state model. Sparkplug 3.0 was published as international standard ISO/IEC 20237:2023, and the Eclipse Foundation continues to steward the specification.[5]

Sparkplug's state model is its most valuable idea. When an edge node connects, it publishes a **birth certificate** listing every metric it will report, with data types and current values. After that it publishes only what changes, an approach Sparkplug calls **report by exception**. The node registers a **death certificate** as its MQTT last will, so if it drops off the network the broker announces it, and every consumer knows the data from that node is stale rather than silently trusting the last value.[6] Sparkplug topics follow a fixed pattern:

`spBv1.0/<group>/<message type>/<edge node>/<device>`

ISA-95: the shape of the namespace

ISA-95, published internationally as IEC 62264, defines a role-based equipment hierarchy that runs from enterprise to site, area, work center, and work unit.[7] It was written to describe how manufacturing operations and business systems relate, and it gives a UNS something it badly needs: a naming structure that is not invented from scratch by

whoever sets up the broker. A topic path built from that hierarchy is readable by an operator, a process engineer, and an ERP analyst alike.

OPC UA: where it fits

OPC UA is not a competitor to a UNS; it is one of its best data sources. Many modern controllers, drives, and machines expose rich, modeled data over OPC UA, and an edge node can read it there and publish it into the namespace. OPC UA also defines its own publish and subscribe model in Part 14 of IEC 62541, which includes a mapping onto MQTT.[8] The practical rule is simple: use OPC UA, Modbus, EtherNet/IP, S7, and the other native protocols to talk to equipment, and use the UNS to share what the equipment said.

STANDARD	PUBLISHED AS	WHAT IT GIVES THE UNS
MQTT 3.1.1	OASIS Standard (2014); ISO/IEC 20922 (2016)	Lightweight publish and subscribe transport, hierarchical topics, retained messages, last will
MQTT 5.0	OASIS Standard (2019)	Everything above plus reason codes, shared subscriptions, message expiry, and user properties
Sparkplug 3.0	ISO/IEC 20237:2023	Defined payloads, birth and death certificates, report by exception, and state awareness
ISA-95	IEC 62264	An equipment hierarchy that gives the namespace a structure the whole business understands
OPC UA	IEC 62541 (Part 14: PubSub)	Rich, modeled data at the machine, and a publish and subscribe mapping onto MQTT

SECTION IV

Designing the Namespace

The broker is the easy part. A broker can be installed in an afternoon. The namespace, the structure of topics and the meaning of what is published to them, is where a UNS either becomes the plant's single source of truth or turns into the same mess it was meant to replace, only faster.

Structure it like the plant

Start from the ISA-95 hierarchy and stop where it stops being useful. A typical topic path reads from the top of the business down to the thing being described:

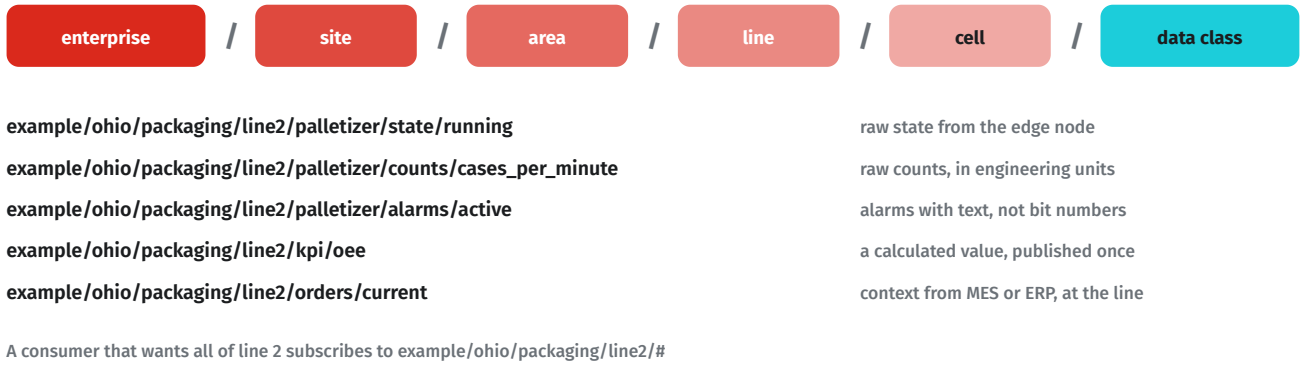

Figure 2. An ISA-95 shaped topic tree. Raw values, calculated KPIs, and business context live side by side under the same line, so a consumer finds all three in one place.

Rules that keep it clean

- **One owner for the structure.** Someone decides what the top levels are and how new branches get added. Without an owner, every integrator invents a convention and the namespace fragments.
- **Plain names, consistent case.** `cases_per_minute`, not `CPM_2`. Pick lowercase or a single case style and never mix them; MQTT topics are case sensitive.
- **Engineering units, not raw counts.** Publish 42.5 degrees Celsius, not register value 4250. Do the scaling once, at the edge, instead of in every consumer.
- **Publish calculations once.** When OEE or a cycle time is calculated, publish the result into the namespace so every dashboard shows the same number.
- **Put business context next to the machine.** The current order, product, and shift published under the line make every machine value meaningful to the business.
- **Decide on Sparkplug, plain MQTT, or both.** Sparkplug gives state awareness and defined payloads, but its topic structure is fixed by group and edge node. Many plants use Sparkplug from the edge and publish contextualized, ISA-95 shaped topics alongside it for business consumers. Either way, decide deliberately and write the decision down.

Mistakes we see most

MISTAKE	WHAT HAPPENS	WHAT TO DO INSTEAD
Publishing every tag	The namespace fills with thousands of unused values, the broker works harder, and nobody can find anything	Publish what the business uses, and add more when someone asks for it
Tag names as topics	Topics like `PLC5/N7/42` carry no meaning and change whenever the PLC does	Map tags to meaningful names at the edge node
No state awareness	A gateway goes offline and dashboards keep showing its last value as if it were live	Use Sparkplug death certificates or an MQTT last will on every edge node
One broker, no plan B	The hub of the plant's data becomes its single point of failure	Cluster the broker or run one per site, and let edge nodes buffer while it recovers
Writes from anywhere	Any client that can publish can change a setpoint	Keep writes to a small, controlled path that the PLC validates (Section V)

SECTION V

Securing the Hub

A Unified Namespace concentrates the plant's data in one place. That is its value, and it is also why it has to be secured as seriously as anything else on the plant floor. The goal is a namespace that every authorized system can use and nothing else can touch.

Treat the broker as a conduit

The ISA/IEC 62443 series describes industrial networks as zones joined by conduits, with controls on every conduit.[9] In that language, the broker is a conduit between the control zone and everything above it. Design it that way:

- **Edge nodes connect out, nothing connects in.** Edge nodes open outbound connections to the broker. No system above the control network needs a path into it, and no inbound port is opened toward the PLCs.
- **Read only by default.** Edge nodes read from equipment. They do not write to it unless a write path has been designed, reviewed, and approved on purpose.
- **Encrypt every connection.** MQTT over TLS for every client, including the ones inside the plant.
- **Authenticate every client.** Each edge node, application, and user gets its own identity, preferably a certificate, never a shared login.
- **Authorize every topic.** Access control lists give each client exactly the branches it needs: an edge node publishes only under its own cell, a dashboard subscribes only to the lines it displays, and almost nothing can publish a command.

Writes are a design decision, not a default

The most common question we get about a UNS is whether it can send setpoints and commands back to the floor. It can, and sometimes it should: a recipe download from the MES, or a production order sent to a line. But every write path is a path into the process, so we build them the same way every time:

1. Writes go to a small set of dedicated command topics, never to the topics that carry state.
2. Only named, authenticated clients may publish to those topics.
3. The edge node passes a command to the PLC as a request, and the PLC program decides whether to accept it, inside the limits it already enforces.
4. Interlocks and safety functions never depend on the namespace. If the broker disappears, the process keeps running safely.
5. Every command is logged with who sent it and when.

> **Plan for the link to fail.** Networks drop, brokers restart, and cloud connections go down. A well-built UNS keeps every line running when that happens: edge nodes buffer locally and publish when the broker returns, and consumers use the death certificate or last will to show stale data as stale.

SECTION VI

Rolling It Out Without Stopping the Line

A Unified Namespace does not need a big bang. The plants that succeed start small, prove value on one line, and let the namespace grow as each new consumer arrives. A rollout we would recommend looks like this:

Crawl: one line, one question

Pick one line and one question the business is already asking, such as why the line misses its rate or where its downtime goes. Install edge nodes next to the existing PLCs, read only, and publish the handful of values that answer the question: running state, counts, alarms, and the current order. Put a dashboard on top. Nothing in the PLC programs changes.

Walk: the business systems subscribe

Once the line's data is trusted, connect the systems that need it. The historian subscribes to store it. The MES subscribes to production counts instead of polling the PLC. Maintenance subscribes to run hours and alarms. Each new consumer is a subscription and a permission, not a project.

Run: more lines, more sites, one structure

Add lines and areas under the same structure. For multiple sites, each site runs its own broker so it stays fully functional on its own, and selected branches are bridged up to an enterprise namespace for corporate reporting. The same topic path that identifies a palletizer on line 2 in Ohio identifies its twin in another plant.

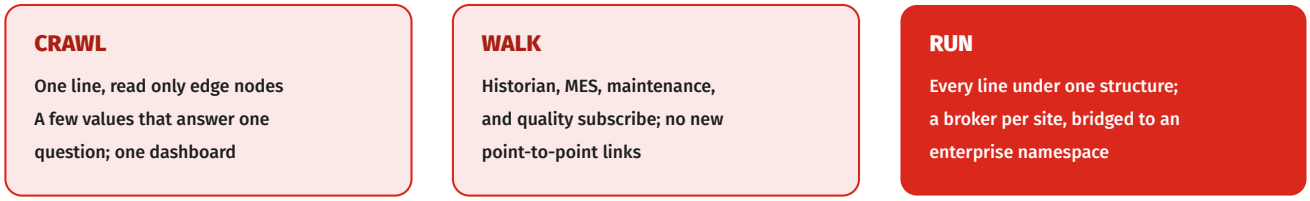

Figure 3. A Unified Namespace grows with its consumers. Each stage pays for itself before the next one starts.

What to measure

Measure the namespace by what it changes, not by how many tags it holds:

- Time to connect a new consumer, from request to live data. In a working UNS this is measured in hours, not weeks.
- The number of point-to-point integrations retired.
- Whether every report on the line shows the same number for the same thing.
- Polling load removed from the PLCs.
- Downtime minutes explained, which the downtime calculator on our Resources page can turn into dollars.

SECTION VII

Building It with Fireball

Fireball Industries designs and builds Unified Namespaces for manufacturers, from the edge node on the first line to the business systems that subscribe at the top. We are vendor agnostic: we work with the PLCs, SCADA, MES, and ERP the plant already runs, including Rockwell Automation, Siemens, CODESYS, and Ignition, and we build on open standards so the namespace belongs to the plant, not to us.

The work covers the whole path:

- **Unified namespace (MQTT)** design: the topic structure, naming rules, and the governance that keeps it clean
- Edge nodes that read existing equipment over its native protocols, read only, publishing outbound
- **HMI, SCADA, and digital commissioning**, so new screens and logic are validated before they reach the floor
- **ERP and MES solutions**, and **cloud deployment and business system integration**, connected as subscribers instead of point-to-point links
- OT visibility and secure data collection. For this work we deploy EmberNet, whose collectors are read only and move data out of the plant outbound only.

How a project runs

1. **Conversation.** What runs on your floor today, what keeps stopping it, and the question you want the data to answer first.
2. **Scope and quote.** The first line, the values that matter, the consumers, and a fixed scope before anyone signs.
3. **Design and build.** The namespace structure, the edge nodes, the broker, and the security model.
4. **Digital commissioning.** The namespace and its consumers validated before they touch the line.
5. **Installation and startup.** Edge nodes installed next to the existing PLCs, with no change to the control programs.
6. **Support.** New lines, new sites, and new consumers added under the same structure, with remote and on-site support from the engineers who built it.

Start with one line.

Tell us which line and which question, and we will show you what the namespace looks like before you commit to the rest of the plant.

sales@fireballz.ai | fireballz.ai/get-a-quote

METHODOLOGY AND REFERENCES

How the Numbers Were Built

Point-to-point links. The number of possible links between n systems is the number of pairs, n(n−1)/2: 10 systems give 45, 20 give 190, and the 8 in Figure 1 give 28. This is a ceiling, not an estimate of any particular plant; it shows how integration effort scales, not what a given site has built. A Unified Namespace needs one connection per system, so 10 systems need 10.

Inbound connections. The zero in the executive summary describes the architecture this paper recommends, in which edge nodes open outbound connections to the broker. It is a design property, not a measurement.

Standards. Every standard and date in this paper is taken from the publishing body's own announcement or listing, cited below.

1. OASIS. *MQTT Version 3.1.1*, OASIS Standard, 29 October 2014. docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html
2. OASIS. "MQTT Internet of Things Standard Now Approved by ISO/IEC JTC1" (ISO/IEC 20922), 19 July 2016. oasis-open.org/?p=3952
3. OASIS. *MQTT Version 5.0*, OASIS Standard, 7 March 2019. docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html
4. HiveMQ. "The History of MQTT Part 1: The Origin." hivemq.com/blog/the-history-of-mqtt-part-1-the-origin
5. Eclipse Foundation. "The Eclipse Foundation Announces Sparkplug as an International Standard," 7 November 2023, and ISO/IEC 20237:2023, *Information technology: Sparkplug version 3.0*. newsroom.eclipse.org/node/39754; iso.org/standard/86204.html
6. Eclipse Foundation. *Sparkplug Specification* (birth and death certificates, report by exception, topic namespace). eclipse.org/tahu/spec/sparkplug_spec.pdf
7. IEC 62264-1:2013, *Enterprise-control system integration, Part 1: Models and terminology* (ISA-95), role-based equipment hierarchy.
8. IEC 62541-14:2020, *OPC Unified Architecture, Part 14: PubSub*, including the MQTT transport mapping. webstore.iec.ch/publication/61108
9. ISA/IEC 62443 series, *Security for industrial automation and control systems*, zones and conduits model.

CONTACT

Talk to an engineer.

Fireball Industries builds standard automation products and the engineering services behind them for manufacturers across the United States and Mexico, from our headquarters in Macedonia, Ohio, our engineering office in Yuma, Arizona, and our operations in Zacatecas, Mexico.

Email
sales@fireballz.ai

Web
fireballz.ai

Get a quote
fireballz.ai/get-a-quote

More white papers
fireballz.ai/resources

© 2026 Fireball Industries LLC. All rights reserved. MQTT, Sparkplug, ISA-95, OPC UA, and the other standards named in this paper belong to their respective standards bodies; product names belong to their owners and are used here only to describe compatibility.