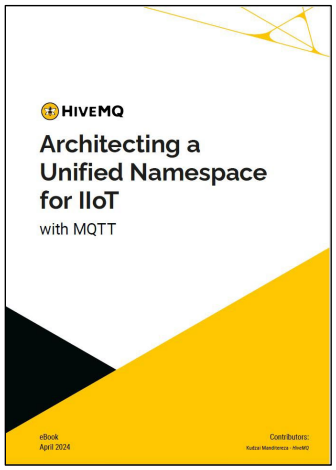


Architecting a Unified Namespace for IIoT

Thank you for downloading this HiveMQ eBook. Carahsoft is the master government aggregator for HiveMQ IoT solutions available via OMNIA, IPHEC, and other contract vehicles.

To learn how to take the next step toward acquiring HiveMQ's solutions, please check out the following resources and information:

For additional resources:
carah.io/hivemq-resources

For upcoming events:
carah.io/hivemq-events

For additional HiveMQ solutions:
carah.io/hivemq-solutions

For additional IoT solutions:
carah.io/IoT-Solutions

To set up a meeting:
HiveMQ@carahsoft.com
571-591-6210

To purchase, check out the contract vehicles available for procurement:
carah.io/hivemq-contracts

Architecting a Unified Namespace for IIoT

with MQTT

eBook
April 2024

Contributors:
Kudzai Manditereza - *HiveMQ*

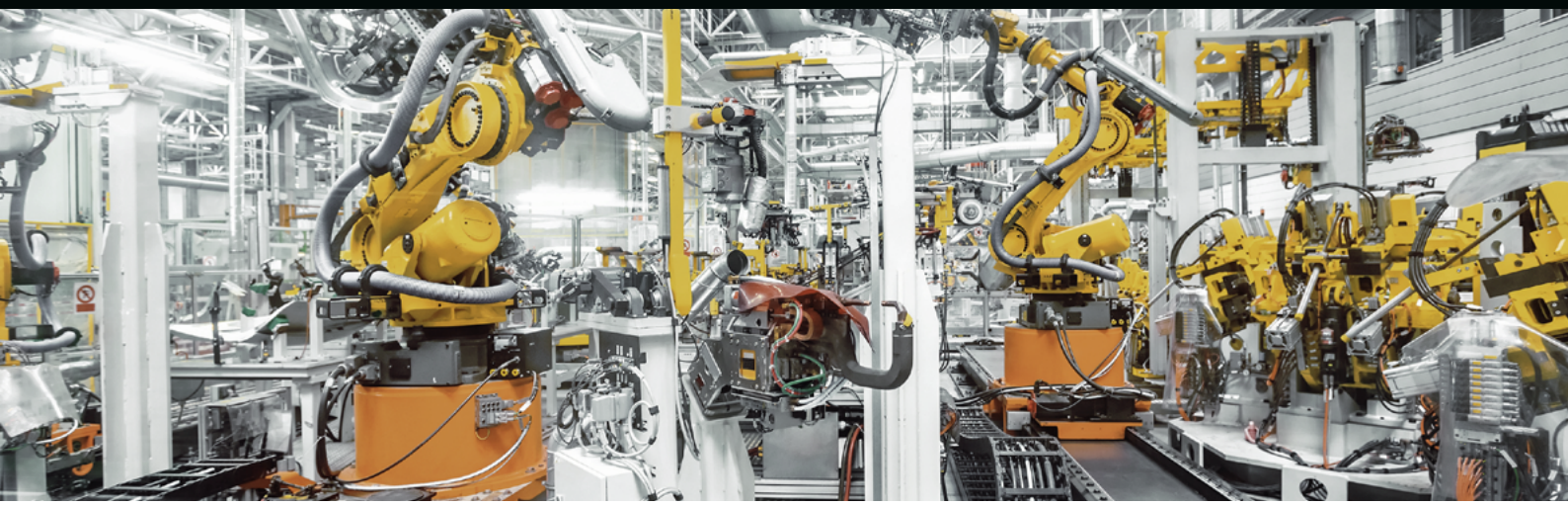


TABLE OF CONTENTS

Foundations of a Unified Namespace for IIoT

In the manufacturing sector, a significant shift is underway. Organizations are transforming themselves to use digital technology not just to support their existing operations but also to drive and transform their business outcomes by augmenting their operations with data-driven intelligence.

However, a persistent challenge remains amidst this evolution. The prevailing view is that to extract value from data, it must be consolidated from various business divisions into a central repository, such as a data warehouse or data lake, and then leveraged to drive an organization's innovation strategy.

This strategy is especially problematic in manufacturing due to the wide variety of Operational Technology (OT) and Information Technology (IT) systems with different meanings and definitions of data. That is, as more data is collected and stored centrally, the inconsistencies make it increasingly difficult to make sense of the data, which is detrimental to advanced analytics that organizations need for intelligent decision-making.

But that's just part of the story: more varying data-generating sources are being connected rapidly across OT and IT domains, and manufacturing businesses are growing in organizational complexity, and with it, the need to address more specific advanced analytics use cases has never been more essential.

To successfully achieve digital transformation on an enterprise scale, manufacturers need an architectural approach that facilitates seamless data integration and the extraction of value, scales effectively with the addition of new data sources and varying contexts, and enables rapid responses to data-driven use cases in manufacturing.

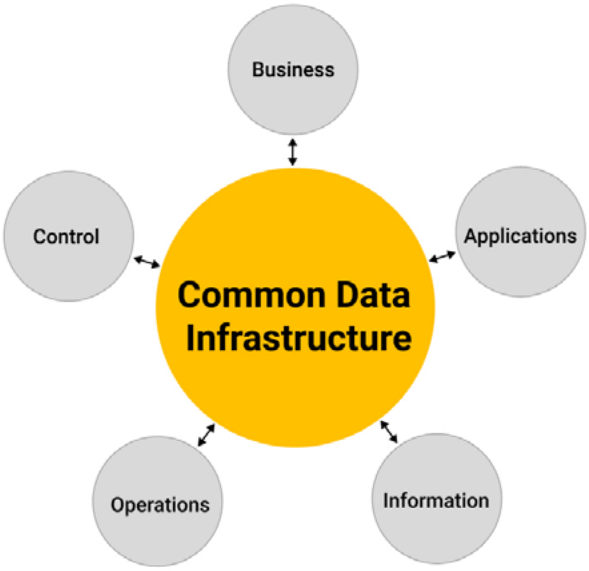
The Unified Namespace (UNS) offers an alternative approach to traditional industrial data architecture. Traditional methods typically follow the ISA-95 pyramidal network-and-system architectural model, which relies on propagating data upwards for centralized storage and subsequent analysis. In contrast, the UNS approach externalizes contextualized data across various functional domains into a real-time semantic hierarchy, establishing a hub and spoke model that serves as a single source of truth for your business's current state and events.

This eBook aims to build upon the principles introduced in a previous guide, UNS Essentials, by offering **practical guidance for architecting a UNS for your organization**. Read on to learn more about UNS, how to design a UNS, data and functional modeling, and how to secure the UNS.

Principles of a Unified Namespace Architecture

Edge-driven and Domain Ownership

The fundamental concept of a UNS architecture is that for a manufacturing enterprise to fully leverage decentralized real-time data sharing, the various components within a specific functional domain must push data from its source at the edge into a common data infrastructure. In manufacturing, these functional domains typically include control, operations, information, applications, and business. The data exchanged may include sensor updates, events, alarms, status changes, commands, and configuration updates. In the edge-driven principle, the data may only be shared when a change has been detected in the monitored item, report-by-exception.

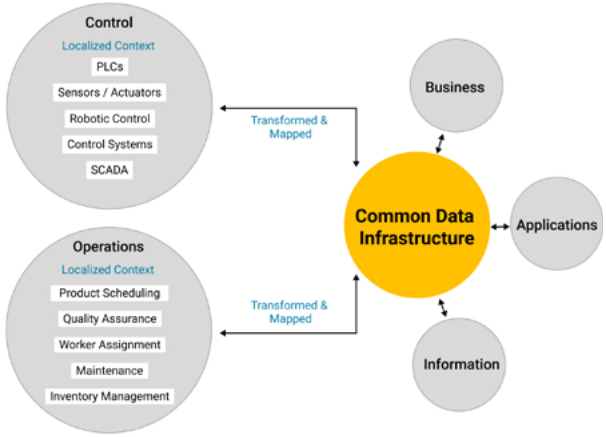


By channeling data straight from the originating systems at the edge into a common data infrastructure — abstaining from the traditional, tiered approach — edge-driven architecture creates the notion of a perpetual, real-time representation of your enterprise's current state and events, effectively creating a singular, authoritative reference point for your business model.

This architecture model therefore serves as a single source of truth, where all components participate as nodes in the ecosystem from which they equally consume information as much as they publish it. Furthermore, an edge-driven architecture enhances reliability by eliminating the fragility of the need for synchronization in systems that rely on request-response interactions. For example, during communication disruptions, this design allows data that would otherwise be missed through request-response models to be stored temporarily and then resent once the network connection is reestablished.

Components in each functional domain in manufacturing are really about one concept of the business, and therefore, share a common interpretation of data semantics. For instance, within the control domain, you find systems like programmable logic controllers (PLCs), supervisory control and data acquisition systems (SCADA), and robotic controls, whose interpretation of data is centered around controlling and monitoring production processes.

As a result, it makes sense to allow each functional domain in your architecture and teams therein a considerable amount of sovereignty in how they package the data, albeit following the guidelines of federated data governance. This autonomy allows the tailoring of data models to address advanced analytics use cases specific to that domain, which empowers regular users, who may not have specialized skills, to analyze data without relying on data scientists to sift through data that has been gathered and organized centrally.

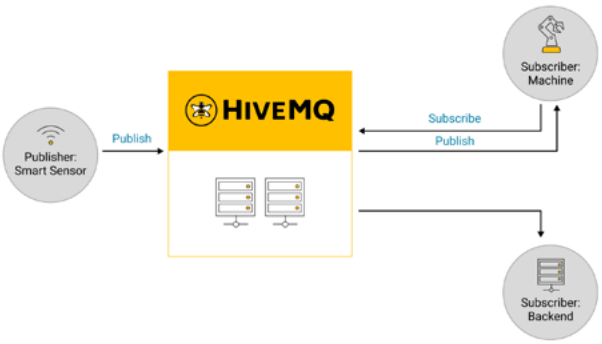


Now the data can be transformed and mapped into an enterprise-wide namespace to ensure consistent and integrated use across different business areas. This approach helps to manage complexity by creating clear interfaces and interactions between different parts of your architecture, which can evolve independently while still integrating with one another as necessary. It enables the effective distribution of data while maintaining flexibility, security, oversight, and separating concerns on an enterprise scale.

Open Architecture and Standard Data Infrastructure

For the UNS to function effectively, it's essential to establish a data infrastructure that supports an open architecture. This means using a standardized method for exchanging information that is openly accessible and widely adopted. Moreover, the infrastructure should incorporate a publish/subscribe model, which enables a flexible and decoupled way of sharing data within a functional domain and across functional domains within your enterprise. This setup must be efficient in terms of bandwidth use and reliability. In addition, a UNS-based system must be self-aware: it should seamlessly integrate new participants and their data into the existing communication network without manual intervention.

The MQTT protocol has emerged as the communication standard in a UNS architecture, as it embodies the core qualities needed for a robust data infrastructure highlighted above. It facilitates the sharing of data — and therefore value — among all entities within the UNS architecture and reduces the complexity and cognitive load in data exchange. You can download our MQTT Essentials Guide to learn more about how MQTT works.



An open architecture plays a crucial role in fostering innovation within a manufacturing organization. It allows teams to formulate hypotheses about data applications and promptly access the necessary tools to test these ideas. With an open architecture, these tools can be effortlessly integrated into the existing data infrastructure, eliminating the need for specialized connectors or converters to handle proprietary interfaces. This streamlined integration accelerates the innovation process, as teams can focus on testing and developing ideas rather than navigating technical compatibility issues.

Moreover, embracing an open architecture avoids the accumulation of technical debt that often comes with custom-built connections for proprietary interfaces. Not being locked into vendor-specific solutions allows the organization to adapt more swiftly to technological advancements. This adaptability means that future integrations, upgrades, or changes can be implemented without the need to overhaul the existing infrastructure or rewrite custom code, saving time and resources and achieving faster time to market.

Federated Data Governance

Let's address a principle designed to bring order and clarity to the distributed, domain-oriented architecture advocated by the UNS: federated data governance. This management system is designed to maintain data quality and ensure interoperability within a distributed domain-oriented environment. The principle behind this approach is to manage and govern data across various domains or business areas, aiming to achieve consistent integrity, interoperability, accessibility, security, and privacy.

This form of governance promotes the development of universal standards, practices, and policies for data handling and use while also honoring the individual governance of different functional domains. It encourages different data custodians across a manufacturing enterprise to collaborate on maintaining data quality and interoperability by establishing common data standards, formats, and protocols.

In the context of UNS, governance must balance the need for localized control and optimization within autonomous domains with the need for overarching harmony across all domains. This balance acknowledges that the system is dynamic and cannot be rigidly controlled. Governance in UNS favors empowering teams close to the data, as they are the most informed and, hence, most capable of managing and sharing their data and ensuring that it is usable, valuable, and feasible to generate.

Core Components of a UNS

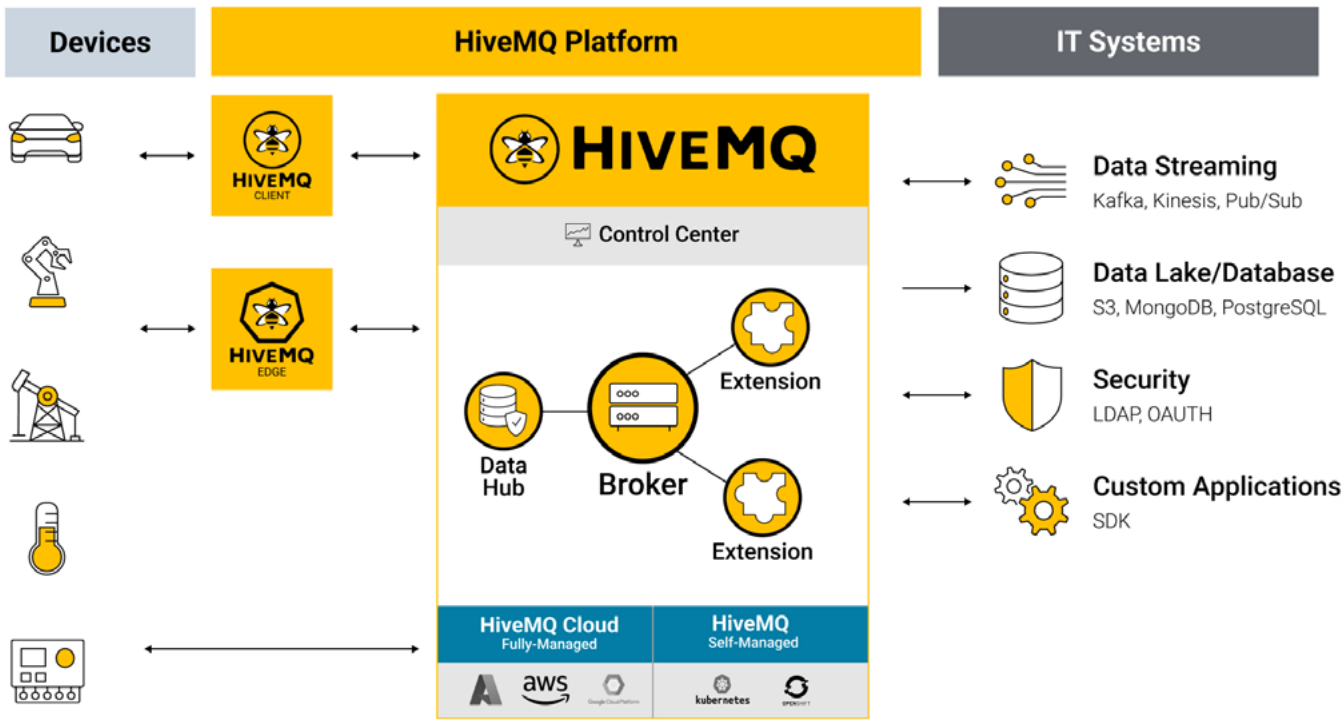
MQTT Broker for the Unified Namespace

At the heart of the UNS architecture is the MQTT broker, a pivotal piece that acts as the central hub for data communication. When planning your UNS, your setup will likely involve multiple MQTT brokers tailored to specific needs within your architecture. These may include robust enterprise MQTT broker clusters for high availability, such as HiveMQ, and machine connectivity solutions with embedded MQTT brokers like HiveMQ Edge.

The critical factor is ensuring that your broker fully adheres to the OASIS standard MQTT specifications to stay true to the UNS principles of an open architecture, which empowers you to select best-in-class tools to plug into your MQTT data infrastructure.

Your architecture requirements will determine which version to choose between MQTT - 3 or 5. Most industrial devices and applications support MQTT 3. However, MQTT version 5 brings advanced capabilities, such as user properties and shared subscriptions, which can streamline certain processes more efficiently than version 3. On the other hand, MQTT 5 includes several optional features, which means a broker can be considered MQTT 5 compliant without supporting every feature. This is significant regarding features like persisting retained messages, which are optional in MQTT 5 but were given in MQTT 3.

For your UNS, retained messages are critical; they ensure that the latest information on any topic is available for new participants, allowing them to access the current state and events within the UNS immediately upon joining without the need to wait, or in the case of MQTT Sparkplug, query devices and applications across the network. We'll discuss MQTT Sparkplug's role in your architecture later in this eBook. When specifying your UNS data infrastructure requirements, it's crucial to define which MQTT features are necessary for



your use case, including which ones the broker must support to fulfill your UNS's goals effectively.

IIoT Platform for the Unified Namespace

In industrial settings, which are often composed of a mix of modern and older equipment with proprietary interfaces, there exists a challenge in integrating this diverse technology into a homogenous data ecosystem. This is where an Industrial Internet of Things (IIoT) platform becomes essential.

It serves as a bridge, connecting older, legacy systems that cannot directly communicate using the MQTT protocol, which is central to a UNS ecosystem. The IIoT platform facilitates the collection of data from these varied sources — ranging from PLCs to SQL databases — and publishes it to the MQTT network.

This platform does more than merely collect and transmit data. It organizes and refines the data by categorizing it, defining its structure and properties, and enhancing its readability and reliability. This process may include aggregating data, performing calculations, or converting data formats, all of which add valuable context that aids in identifying patterns, trends, and anomalies. This ensures that the data is not only accessible and understandable within its local domain but is also prepared for integration and analysis across different functional domains.

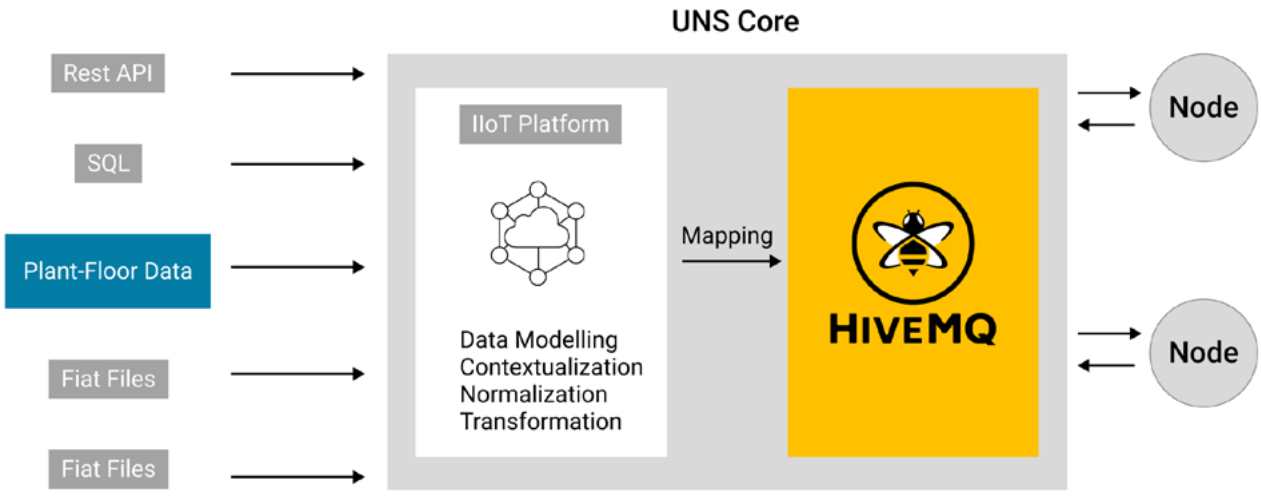
At its core, the UNS is the combination of an MQTT broker and an IIoT platform. The underlying principle is straightforward: any new data that is generated behind the IIoT platform using point-to-point should be published to the UNS, ensuring that it's shared across the network as the current state and events. Additionally, current data might be a snapshot of historical records pertinent to the present moment, but ultimately, the approach taken should be tailored to the problem.

Users can bypass the UNS and access data directly from the IIoT platform for localized or immediate data requirements. While direct connections, like linking a historian to a SCADA system, are sometimes beneficial, the decision to use such connections depends on the specific issue.

Data Persistence for the Unified Namespace

For an effective UNS architecture, relying solely on an MQTT broker's real-time data-sharing capabilities and an IIoT platform is insufficient. Data persistence is crucial, meaning you need the ability to store and access historical data. This requires incorporating both a historian (or time-series database) and a structured database, such as SQL, into your UNS architecture.

The historian plays a pivotal role by subscribing to the UNS and archiving data over time, allowing for retrospective analysis and insight. This historical data should be easily accessible through the same IIoT platform managing the UNS.



Additionally, the SQL database, typically integrated with the IIoT platform, holds structured data essential for operational management and analysis.

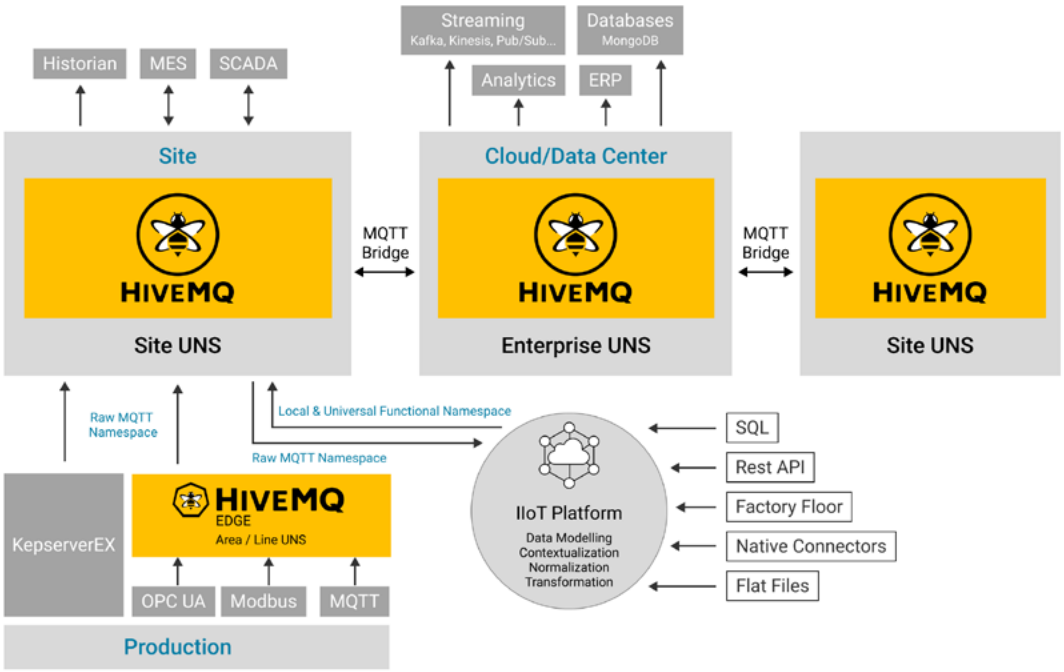
The historian mirrors the live data model of the UNS but with the key difference of maintaining a historical record. From there, REST endpoints can be established to query historical data, and these queries are made relevant within the UNS by publishing them back to an appropriate level of the semantic hierarchy.

Furthermore, transactional operations on stored data may be facilitated through the IIoT platform acting as a bridge between the UNS and a data store. For example, a machine's downtime event captured by the MQTT broker can prompt an update to an SQL table used for Manufacturing Execution Systems (MES). The MES might use historical SQL data to calculate current Key Performance Indicators (KPIs), which are then fed back to the MQTT broker. Similarly, current work on a machine could be informed by historical data on similar past work orders, allowing operators to compare and improve performance based on this historical context. These comparisons can be published back to the MQTT broker, effectively bringing historical performance benchmarks into the present analysis.

For recurring data queries, it's more efficient to directly connect to the historian through the IIoT platform or set up specific request/response topics within the UNS. However, the UNS should not be used as a primary API for ad-hoc data queries. Instead, queries should be directed through the native tools that manage the databases, ensuring efficiency and effectiveness.

Unified Namespace Reference Architecture with MQTT

Below is the UNS Reference Architecture based on the core components discussed. Again, as you can see, the MQTT broker and IIoT platform play a significant role in architecting a Unified Namespace.



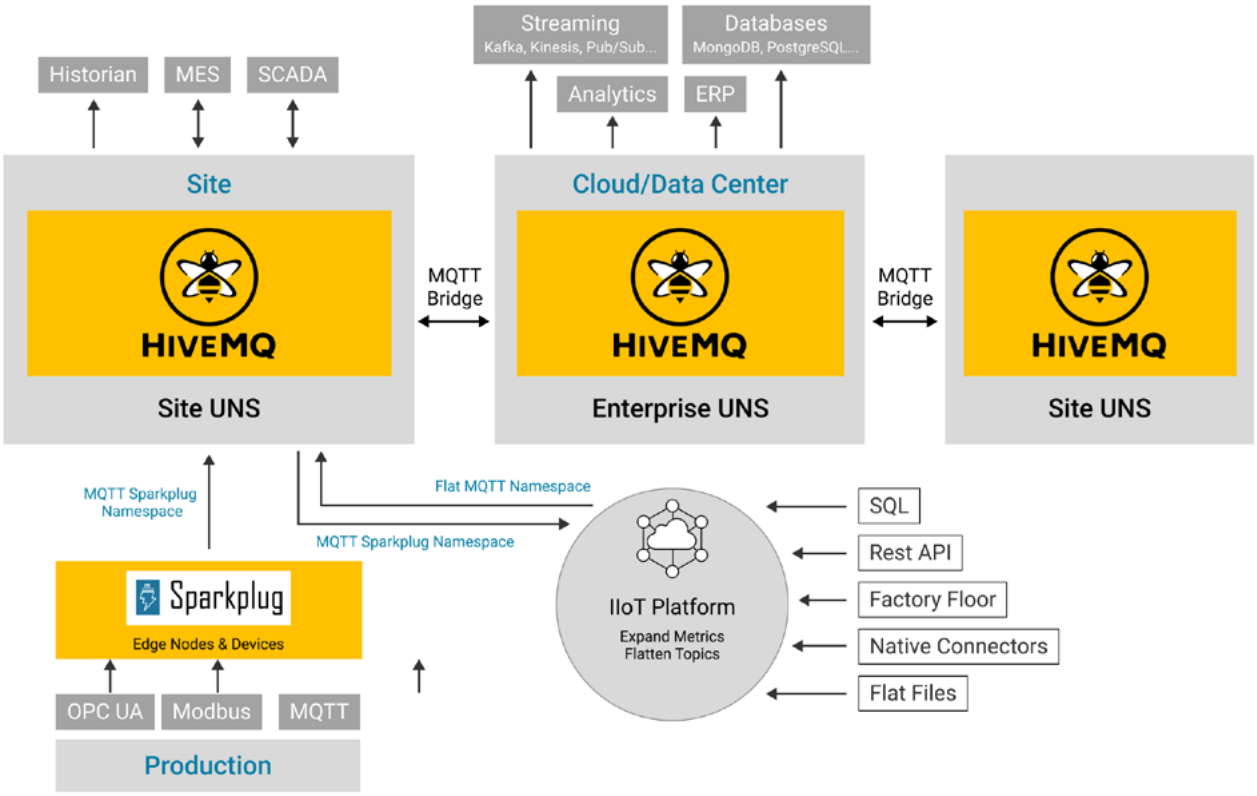
The reference architecture also includes HiveMQ Edge to facilitate data conversion from point-to-point protocols to MQTT for simplified integration. HiveMQ Edge also embeds an MQTT broker, which allows the data to be mapped into a local namespace within a production area or line.

Data can also be gathered from devices with MQTT capabilities and connectivity solutions like KepserverEX. This data is then fed into a dedicated "raw MQTT namespace." From there, the IIoT platform processes this data to give it context and a consistent format before redistributing it into the UNS.

The use of MQTT bridges enables the multiple MQTT brokers and different levels of a manufacturing enterprise to share information, creating a distributed network of data brokers that ensures data consistency and availability across the network. This is crucial for large-scale, geographically distributed industrial environments.

Unified Namespace Reference Architecture with MQTT Sparkplug

MQTT Sparkplug offers additional interoperability within the MQTT network. It standardizes MQTT Topic Namespace, State Management, and Payload Structure, offering benefits for an enterprise UNS architecture. MQTT Sparkplug can be used up to Level 2, where a Sparkplug consumer, typically an IIoT platform, can be used to subscribe to the Sparkplug namespace and perform some transformation that could make it easier to integrate data into enterprise systems that are not Sparkplug compliant.



UNS represents a groundbreaking architectural approach to digital transformation in the manufacturing sector. Organizations can effectively manage complexity and enhance data interoperability by adopting an edge-driven architecture and leveraging an open architecture that utilizes a standard data infrastructure like MQTT. In the next section, we will discuss how to design your UNS semantic hierarchy with information mapping.

Designing Your UNS Semantic Information Hierarchy

Now that we have explored the fundamentals of a UNS architecture for IIoT, we will discuss UNS semantic data hierarchy design using MQTT.

To succeed in digital transformation, manufacturers should embrace an architecture that ensures seamless data access and integration across diverse platforms and applications within specific areas and throughout the organization. Establishing a clear semantic hierarchy, which includes layers of structured data, is crucial for intuitive and open access to this data. This is essental for leveraging data effectively to enhance intelligence in manufacturing operations.

Structured data is important for several reasons, including:

- Analysis and insight: Structured data is more easily analyzed, meaning businesses can gain actionable insights quicker, leading to improved decision-making and process optimization.
- Machine learning and AI: Structured data is essential for training machine learning models and deploying AI solutions, which can lead to further automation and efficiency gains.
- Real-time processing: IIoT often requires real-time data processing. Structured information can be processed more quickly, which is critical for real-time applications.
- User experience: Well-structured data can improve the user experience by providing easier access to relevant information and more intuitive interfaces.
- Interoperability: Structured information ensures that data from different sources can be integrated and used across different platforms and applications.
- Scalability: As businesses grow, so does the amount of data. Structured information ensures that systems can scale without losing performance or data integrity.
- Data Management: Structuring data helps in organizing, storing, and managing it efficiently, enabling faster access and analysis.

Therefore, the UNS architectural approach offers a real-time, semantically-organized hierarchical data structure. This structure acts as a central hub, a single source of truth, reflecting a business's current state and events. It grants every network participant immediate access to information throughout the manufacturing organization and provides a clear pathway to find analytical data relevant to their roles.

Let us explore how the MQTT protocol can be used to build a semantic data hierarchy for your UNS.

UNS Semantic Data Hierarchy Design Using MQTT

In an MQTT-based publish-subscribe system, the MQTT broker organizes data using a topic hierarchy, which acts as a structured framework for data access. This hierarchical organization enables precise control over data sources within a UNS. Participants within the network can efficiently access required data by subscribing to specific levels of this hierarchy. Additionally, they can use wildcard characters such as '+' and '#' in their subscriptions to receive messages from multiple topics simultaneously instead of subscribing to each topic separately. This structure streamlines the process of data distribution and access within the network.

The structured nature of MQTT topics enables the creation of a comprehensive data access hierarchy within an organization: the UNS hierarchy. This hierarchy comprises numerous namespaces from various data sources arranged under the main topic namespace. Some namespaces may hold unprocessed data from devices and applications while others may contain data that is processed, normalized, and contextualized for local use or for feeding into external systems for further analysis. Keeping a consistent topic structure is essential for a predictable and organized namespace.

```
- manufacturing/
  - plantA/
    - sensors/
      - humidity/
        - sensor001
        - sensor002
        - ...
    - inventory/
      - raw_materials/
        - current_stock
        - reorder_levels
      - finished_goods/
        - current_stock
        - shipping_schedule
    - quality_control/
      - inspection/
        - results
        - trends
      - testing/
        - test001/
          - results
          - next_schedule
          - ....
  - plantB/
        - ...
```

Above is an oversimplified example of the UNS structure to give you an idea. Here, **manufacturing/** is the root topic and represents the entire scope of the organization's manufacturing data. **plantA/** and **plantB/** are sub-topics under the root, representing different manufacturing plants within the organization. Under each plant, there are further subdivisions such as **sensors/**, **machines/**, **robotics/**, **inventory/**, and **quality_control/** which categorize the types of data and operational areas.

This structure allows participants in the MQTT network to subscribe to specific levels of data as needed. For example, a supervisor might subscribe to **manufacturing/plantA/machines/+/#** to receive all messages related to machines in Plant A, while a quality control analyst might only subscribe to **manufacturing/plantB/quality_control/testing/#** to receive data on all tests conducted in plant B.

Regardless of the specific setup, it's essential to establish and document the categories and relationships within your UNS beforehand. The organization's semantic structure should be embedded within the topic namespace, ensuring that information about data access is available independently of the actual data content. While the UNS operates as a decentralized and dynamic system where individual constituent namespaces may alter or cease to exist, the primary namespace structure should stay constant, only changing when transitioning to a new version.

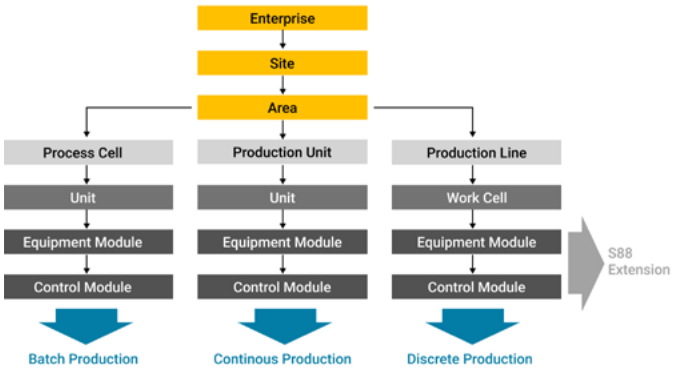
Best Practices for MQTT Topic Namespace Structuring for UNS

ISA-95 Common Data Model

When organizing your MQTT topic namespace for the UNS, it's largely up to the system architect to determine the layout. However, following best practices, such as adopting the ISA-95 common data model, can prove beneficial. This model helps you reflect your organizational hierarchy within the MQTT topic structure, effectively organizing data streams between producers and consumers in a manufacturing environment.

The hierarchy typically follows this pattern:

Enterprise > Site > Area > Line > Cell



Applying this model, your primary MQTT topic namespace for the UNS might be structured like this:

FreshDairy/Munich/Packaging/Line1/Cell1

Versioning Your UNS MQTT Topic Namespace

As your UNS systems develop, incorporating new features might necessitate alterations in the MQTT topic structure. Versioning these topic structures enables developers and operators to handle updates more efficiently, reducing the risk of system failures due to incompatible updates and providing a better experience for end-users by maintaining service continuity. This approach allows older clients to operate reliably on their existing topic structures without the immediate need for updates.

Moreover, versioning accommodates the diverse capabilities and feature support across different clients by allowing them to interact with the specific topic structure version they are compatible with. Developers can leverage newer versions for testing purposes, enabling them to refine changes in isolation from the live system. Additionally, versioned topic structures simplify documentation and maintenance, providing a clear framework for systematic change tracking and management.

Here's an example of creating a versioned MQTT topic namespace that incorporates both your specific naming convention ('spec') and a version identifier ('version'). In this structure, you also include the 'Client ID' within the topic path itself.

Pattern for Versioned Topic Namespace:

<specification-name>/<version-number>/.../$area/$line/$cell/$client-id

For instance, if your specification is named 'mySpec' and you are working with version 1, the topic namespace might be:

mySpec/v1/.../$area/$line/$cell/$client-id

Designing Unified Namespace Data Structure

After defining the primary MQTT topic namespace, you'll need to devise a scheme to organize raw and processed data across your hierarchical structure systematically. Each level should have its own detailed namespaces essential for defining your UNS data architecture. The organization of these namespaces affects the distribution and accessibility of analytical data within the functional domain of its originator (data producer) and how it is packaged for cross-domain integration and actionable analysis.

To achieve this, begin with a strategic plan that identifies the specific namespaces within your UNS and standardize them. Consider the different types of namespaces that will be integrated into your UNS, ensuring they facilitate clear and efficient data communication and analysis across your organization.

Establishing an Edge Namespace for All Your UNS Data

The foundational concept of the UNS architecture is to avoid preconceptions about the future utility of data, recognizing that the value and relevance of data evolve as digital transformation progresses. In alignment with this philosophy, it's recommended to map every piece of data — especially PLC tags — to the UNS. This includes maintaining a distinct namespace for raw data, which you might refer to as the 'edge' or 'raw' namespace. The importance of consistent documentation and implementation of this namespace across your organization cannot be overstated, as it should align with your data governance strategy.

The structure of the topic for this edge namespace would follow the format:

Enterprise/site/area/line/cell/edge/

Within this namespace, you will find models representing physical assets or equipment. These models primarily consist of tags from your PLCs or control systems, capturing the unprocessed, dynamic data streams from your machinery and processes. If you have multiple PLCs in your production line, then you can split up your edge namespace by PLCs. The raw data namespace is crucial as it allows you to start collecting data from day one and have it available in your UNS while you still figure out how to package it for external consumption.

Establishing a Functional Namespace for Your UNS

After collecting raw data into an edge namespace, the next

step is to process this data, making it suitable for various applications within a specific functional domain and across your enterprise. This involves processes such as modeling, normalizing, contextualizing, transforming, and implementing uniform naming conventions. Once processed, this data should be kept on a separate namespace, known as the functional namespace.

The functional namespace is organized in a hierarchical format, typically as follows:

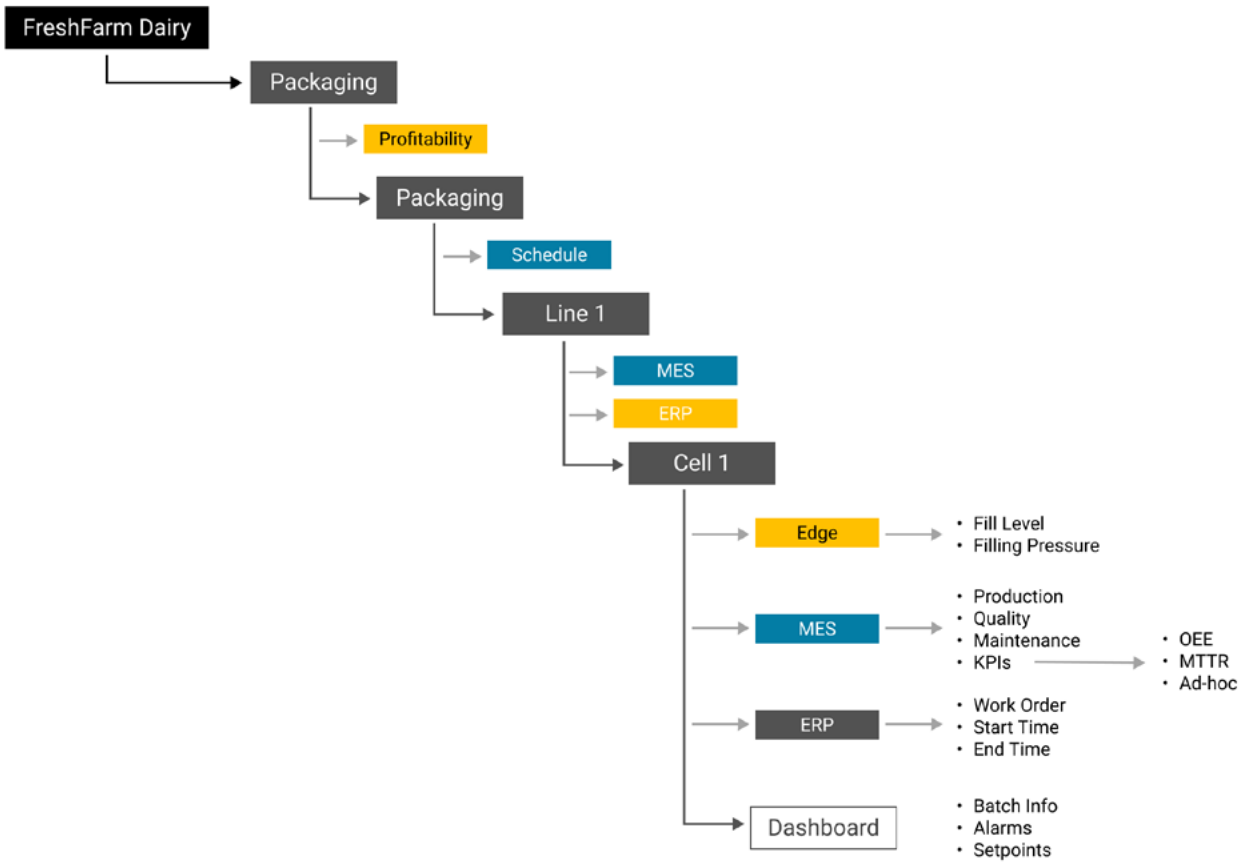
Site/area/line/cell/functions/…

As discussed previously, an IIoT platform is essential for the creation of a functional namespace. It is used to subscribe to data points ('tags') within the edge namespace, apply context and modeling, and then republish the processed data into the functional namespace of your UNS.

This structured approach allows for the creation of functions and visualizations based on the standardized data within the UNS. For instance, an Overall Equipment Effectiveness (OEE) namespace might calculate OEE values from the edge data and present this in a dedicated OEE functional namespace. Ideally, your functional namespace should encompass various KPIs related to production, quality control, and maintenance, like OEE, first pass yield, and mean time to repair.

Besides these universal KPIs, you might have ad-hoc functions tailored to address specific analytical challenges within a particular domain, such as a production line, area, or site. Functional namespaces can also be published and consumed at different organizational levels, such as profit and loss figures at the site level or batch records at the line level.

Below is an example of a UNS semantic hierarchy to give you an idea of what it would look like.



The number and granularity of the namespaces to incorporate into your UNS are subject to the architect's discretion. You might also consider creating informative namespaces, which contain data abstracted for software, data lakes, and other systems, and definitional namespaces, which hold rarely changing parameters like installation dates, calibration records, etc.

As your understanding of data mapping and modeling evolves, the functional namespace will expand. Therefore, it is a continuous process of refinement to ensure the data remains clear, interoperable, and, most importantly, useful.

It's crucial to recognize that while the UNS is a recommended approach, it isn't a one-size-fits-all solution. Different organizations will naturally develop their own unique UNS data structures. The essential principle is to arrange the namespaces within the UNS in a way that mirrors the organizational structure. This alignment ensures that the data is logically organized, making it straightforward for both users and systems to find and access the necessary information.

Additionally, it is of utmost importance to meticulously document the specifications of your namespaces. This documentation ensures uniformity in how the namespaces are implemented within your organization and is vital when sharing data with external partners to guarantee clarity, consistency, and the ability to work together seamlessly.

Removing Inactive Elements from Unified Namespace

Finally, given the dynamic and expansive nature of the UNS, it is common for certain elements, such as devices, sensors, and namespaces originally established for temporary issues, to become obsolete. These elements might linger in the UNS even after their data sources or the problems they were meant to address no longer exist. It's therefore essential to regularly audit isa-95and clean your UNS to eliminate these inactive components.

If you are adhering to best practices by setting the 'retain' flag to true when publishing to your UNS, you can simply publish an empty payload. This action will effectively remove the inactive parts from the UNS.

The Role of MQTT Sparkplug in UNS Data Structure

Sparkplug enhances MQTT networks by providing a standardized format for topic namespaces, which helps organize and identify devices and nodes within the network. This standardization simplifies the process of data discovery across the network.

Below is the standardized Sparkplug topic namespace where **group_id** identifies a logical grouping of Edge Nodes, **edge-node_id** uniquely identifies a specific edge node within the same group, and **device_id** identifies a specific device controlled by the edge node.

spBv1.0/group_id/message_type/edge_node_id/[device_id]

You can learn more about Sparkplug from our Sparkplug Essentials Series.

However, Sparkplug's design presents two main challenges for creating a semantic data hierarchy in enterprise Unified Namespace (UNS) systems. First, the topic namespace in Sparkplug is inflexible, with only three levels for addressing. This limitation can hinder the representation of complex organizational structures. Second, Sparkplug messages contain multiple values, which prevents subscribing to individual data points within these messages.

Despite these challenges, MQTT Sparkplug is popular for its dynamic capabilities and efficient device management in the controls domain. To overcome its limitations, many architects use a combination of Sparkplug and flat MQTT. The recommended practice is to use Sparkplug for the controls domain, particularly for real-time SCADA implementations, and then switch to flat MQTT for higher levels or cross-domain integration.

One method to achieve this is by incorporating the entire semantic hierarchy into the group_id using delimiters within the controls domain and then translating that into a flat MQTT

topic namespace for broader UNS integration. For example:

spBv1.0/Plant1:Area3:Line4:Cell2/# becomes /Plant1/Area3/Line4/Cell2/#

This transition can be handled by an IIoT platform, MQTT broker extensions, or custom programming. The chosen method will also be used to separate Sparkplug's combined metrics into individual MQTT topics for enterprise-level applications.

Ultimately, the decision to integrate MQTT Sparkplug into a UNS architecture or to use flat MQTT exclusively rests with the discretion of the system architect.

In conclusion, the practical implementation of a UNS through an MQTT-based semantic data hierarchy is a transformative strategy for digital manufacturing. It enables seamless data integration and real-time access, thereby fostering a more agile, intelligent, and efficient manufacturing environment. The hierarchical structure of MQTT topics facilitates precision in data distribution and offers flexibility for scale. By adhering to best practices such as the ISA-95 model and incorporating versioning, manufacturers can ensure a robust and scalable infrastructure that aligns with their organizational needs.

The differentiation between edge and functional namespaces is a critical step in managing the lifecycle of data from its raw state to a context-rich, actionable format. Regular auditing and cleansing of the UNS ensure its relevance and efficiency, preventing data obsolescence from hindering operational capabilities. As manufacturers journey through digital transformation, the UNS becomes more than just a technical architecture; it embodies the confluence of data governance, system interoperability, and organizational knowledge, ultimately defining the backbone of a smart manufacturing enterprise.

Data and Functional Modeling for Unified Namespace

In a UNS architecture, different applications and subsystems must understand the data they exchange. This understanding is achieved through a data model, which defines the data's context and meaning, enabling semantic interoperability. Data modeling is, therefore, essential at the start of any data integration process within the UNS. Its primary goal is to outline the structure, relationships, and properties of the data to be used.

Data modeling requires identifying and organizing different types of data attributes. These range from simple operational data to more detailed explanations that define the data's characteristics. For example, these descriptions can include measurement units and other details that together represent something in the real world, like a piece of industrial equipment or a process. A well-structured data model is crucial for organizing data in a clear, understandable manner, preparing it for further processing and analysis. Such structuring is vital for enabling data-driven applications and use cases.

Let's discuss the process of designing your data models for effective integration into the UNS ecosystem. We will also cover the methods for functional modeling that enable interactions within UNS and explain why DataOps is essential for your UNS architecture.

Designing a Data Model for the Unified Namespace

When designing a data model for the UNS, it's crucial to balance robustness, adaptability, and seamless integration with diverse systems and applications. Key properties must be prioritized to enhance the model's functionality, usability, and relevance amidst evolving industrial demands and technological progress.

In UNS, data models are typically established at the edge and fall into two categories: those tailored for local analytical purposes and those standardized for wider enterprise analysis. This distinction should be considered in your data model design.

Another significant aspect to consider is the balance between complexity and data richness. While enriching your model with extensive data semantics is beneficial, it's essential to maintain simplicity in your data payload. This approach aligns with the UNS principle of lightweight data models.

Below is a list of important attributes to consider when designing your data model for UNS. While not exhaustive, this list serves as a foundational guide for planning your data model in the UNS context.

- Model versioning: Your data model must include a version property, similar to the versioning in your topic namespace definition, allowing for tracking changes and ensuring compatibility with different versions of applications.
- Unique identifier: Every object in your UNS system should have a unique identifier, like a serial number or UUID, for precise identification and tracking across various systems.
- Source system identification: In addition to reflecting the source system in the UNS topic namespace, it is often necessary to embed the MQTT topic path within the data model, which ensures clarity without relying on external topic path context.
- Sensor data: This is the core element of UNS objects and varies based on the application, including metrics such as temperature, pressure, humidity, and vibration. Sensor data is vital for analytics and decision-making.
- Descriptive metadata: Basic information about an asset (like a machine), including its type, manufacturer, model, and specifications provides essential context for understanding the data generated by the machine.
- Status information: Real-time data about the operational state of an asset, including error codes and maintenance alerts is key for monitoring the asset's health and performance.
- Operational parameters: Thes are critical settings that determine how a machine operates, such as temperature ranges, pressure levels, or rotational speeds. They can also include work order, product type, and operator ID, varying with the use case.
- Data quality indicators: Details regarding the reliability and accuracy of the reported data, crucial for analytics applications to assess the data's trustworthiness.
- Location information: Integrating direct location data, like GPS coordinates or site names, into the object model, avoiding dependence on external relational structures.
- Timestamps: Implementing a consistent timestamp format for all data points is essential for time-series analysis and critical for applications in analytics and machine learning.

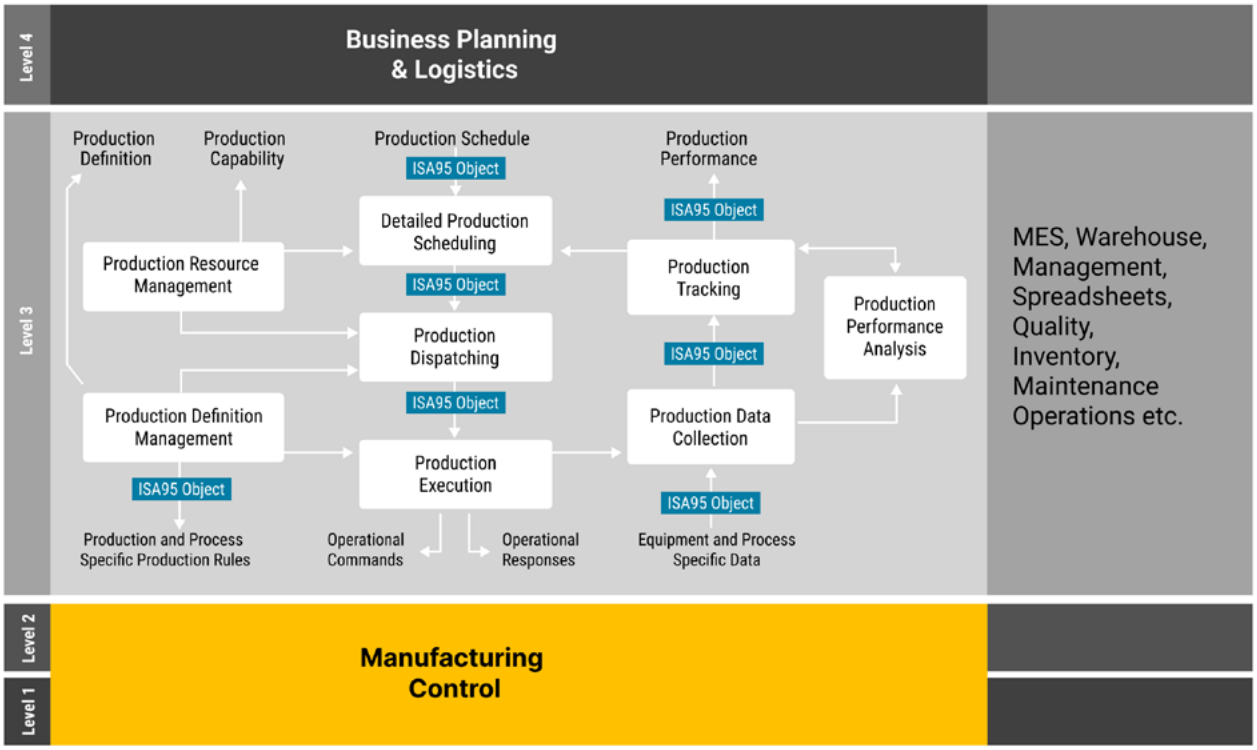
Below is a JSON structure that provides a comprehensive view of a typical UNS object, including all the key aspects required for effective data management and analysis.

```
{
      "modelVersion": "1.0",
      "deviceId": "UUID-1234-5678",
      "sourceSystemId": "Sitel/Area/Line1/Cell1",
      "sensorData": {
        "temperature": 22.5,
        "pressure": 101.3,
        "humidity": 45,
        "vibration": 0.02
    },
      "metadata": {
        "type": "HVAC System",
        "manufacturer": "CoolAir Inc.",
        "model": "CA9000"
        "specifications": {
          "maxTemp": 30,
          "minTemp": 15,
          "maxPressure": 150
       }
      },
      "status": {
        "operationalState": "Running",
        "errorCodes": ["E01", "E03"],
        "maintenanceAlerts": ["Filter Replacement Due"]
      },
      "operationalParams": {
        "setTemp": 21,
        "setPressure": 100,
        "workOrder": "WO-20231201"
        "productType": "TypeA"
        "operatorId": "OP-1001"
      },
      "dataQuality": {
        "temperatureAccuracy": 0.5,
        "pressureAccuracy": 1.0
      },
      "location": {
        "gpsCoordinates": "35.6895° N, 139.6917° E",
        "siteName": "Munich Facility"
      },
        "timestamp": "2023-12-01T12:30:00Z"
```

Designing a Functional Model for the Unified Namespace

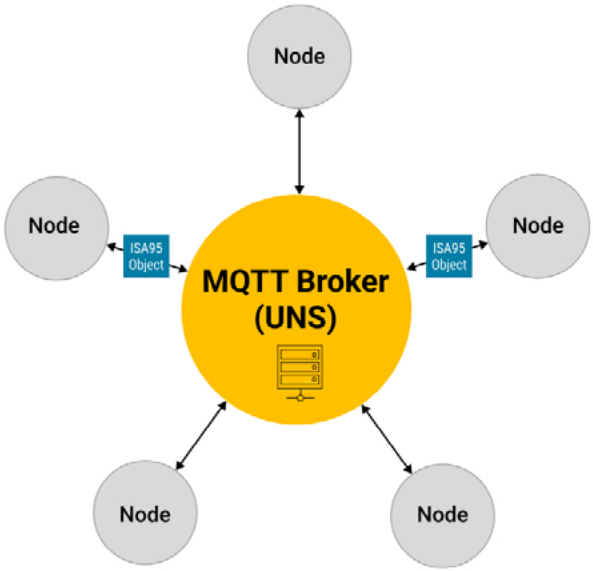
The UNS does more than just represent data objects for physical machines on the shop floor. It also includes transient objects created from activities occurring at Level 3, which is the intermediate layer between the top floor and shop floor. These activities involve coordinating personnel, equipment, and materials to complete tasks. Additionally, they generate data objects that need a clear and consistent model for integration into the UNS system.

When developing data models for the UNS to enable information flow between OT and IT domains, there are two main strategies: build from scratch or use the ISA-95 functional modeling standard as a guide or source of inspiration. The aim of ISA-95 functional data modeling is to identify common patterns in various manufacturing activities highlighted previously, e.g. production tracking, dispatching, and execution. These common patterns are then modelled as data objects to facilitate information exchange in a standardized and uniform way.



Of significance to the UNS architecture is the fact that data objects defined by ISA-95 are implementation-independent. In other words, the ISA-95 is an abstract specification that doesn't describe data types or information models that can be used to create these objects. Also, it doesn't specify what transportation mechanism is to be used to move these objects around. It simply defines a standard terminology for these activities and uses UML representation for the data objects.

This flexibility means that the definitions of these objects can be effectively used in a UNS architecture that utilizes MQTT for data exchange. The choice of how to implement and adapt these standards rests with the system architect. The advantage of such standards is their adaptability; they can be selectively applied and tailored to fit the unique performance and business requirements of a particular architecture.



DataOps for the Unified Namespace

Data modeling is a crucial initial step in establishing the structure, relationships, and characteristics of data objects in your UNS. However, the challenge arises because not all data producers can shape their data to match the specific structure and semantics required by the UNS, and likewise, not all data consumers can efficiently process data formatted in this specific manner. Consequently, data often requires adjustments to suit the distinct needs of various systems. DataOps addresses this issue by developing an abstraction layer to ensure data aligns with the naming and structural standards of the UNS, rather than conforming to the unique formats of its originating systems.

Let's explore the capabilities that DataOps brings to the UNS.

- Data normalization: This process involves refining and standardizing data to reduce redundancy and improve data integrity. In manufacturing plants with data from various sensors and devices, normalization ensures that data is uniform in terms of units and formats, facilitating accurate analysis and efficient storage.
- Data transformation: DataOps is responsible for converting raw data into more suitable formats for analysis. This could involve aggregating data, performing calculations, or transforming it into a more comprehensible form. For example, converting raw speed data from various machines into an average speed per shift allows for easier interpretation and comparison.
- Data contextualization: This aspect of DataOps adds layers of meaning to data by incorporating relevant contextual information. It enhances decision-making by including additional details like product information, maintenance records, and environmental factors. The use of metadata and data lineage in this process enriches the dataset, enabling real-time contextualization for prompt and informed decision-making.
- Iterative data modeling: DataOps promotes a flexible approach to data modeling, encouraging continuous refinement and adaptation based on evolving needs or new insights. This adaptability is crucial for keeping the data model relevant and effective over time.
- Dynamic version control: Maintaining consistency in data models is a challenge in UNS environments, particularly those spanning multiple sites. DataOps introduces dynamic version control, where data models are continuously updated and integrated through CI/CD pipelines. This approach helps prevent conflicts between different versions of data models and ensures they are always up-to-date with the latest requirements.

Essentially, DataOps ensures that data is well-organized, understandable, and optimized for effective application in data-driven use cases.

We've explored the process of designing data and functional models in a UNS architecture. Establishing a robust data model is critical for ensuring semantic interoperability among diverse systems, facilitating data-driven decision-making and operational efficiency. We also discussed the crucial role of DataOps in the UNS ecosystem. By integrating these practices, organizations can significantly enhance their ability to manage and utilize data effectively within a UNS, ultimately leading to improved operational insights and strategic decisions in the industrial domain.

Securing the Unified Namespace Architecture for IIoT

Industrial IoT organizations prioritize the security of their data, and the UNS can allow for maximum security practices. Let's discuss the security challenges associated with UNS in IIoT environments and offer actionable strategies and best practices to address these challenges effectively.

Integrating diverse systems within the UNS architecture inherently increases its exposure to various security threats. These threats pose risks to data integrity and privacy and can lead to operational disruptions and substantial financial losses. In an era where cyberattacks are becoming more sophisticated and frequent, particularly targeting critical infrastructure, the need for robust security measures within UNS environments has never been more urgent.

Let's delve into the security challenges associated with UNS and examine strategies and best practices to address them effectively. From fundamental aspects like authentication and authorization to advanced considerations such as encryption, client management, and data governance, we will explore the various facets of securing a UNS.

Authentication and Authorization

To secure MQTT communications in your UNS, authorization for each client is crucial to prevent unrestricted access to all topics. The MQTT 3.1.1 specification acknowledges the need for authorization in hostile environments.

Topic permissions set by the broker dictate what clients can publish or subscribe to, including topic specificity, operation type (publish, subscribe, or both), and quality of service level. If unauthorized, a client's attempt to publish can lead to disconnection or non-delivery of the message without notification. For subscriptions, the broker can deny access and notify the client if they lack permission for a specific topic. Key strategies include:

- Username and password: The MQTT protocol provides username and password fields for authenticating message exchange. The client can send a username and password when it connects to an MQTT broker.
- Role-based access control (RBAC): This involves assigning permissions based on roles within the organization, ensuring that only authorized users have access to specific MQTT topics.
- Access control lists (ACLs): ACLs provide a more granular level of control, specifying which clients can publish or subscribe to certain topics.
- Integration of OAuth2: Although challenging, using OAuth2 for MQTT broker authentication centralizes the management of access, allowing for more streamlined and secure control of user permissions.

Encryption and Secure Communication

MQTT operates over TCP, which by default, is unencrypted. Many MQTT brokers, including HiveMQ, support TLS as a substitute for plain TCP for secure UNS communication. This is especially important when using MQTT CONNECT packet's username and password for authentication and authorization, to ensure data security. Port 8883 is the standardized, reserved port for MQTT over TLS, known as "secure-mqtt", ensuring exclusive and secure MQTT communications. Key strategies include:

- Transport Layer Security (TLS/SSL): This protocol encrypts data in transit, protecting it from interception and tampering.
- Secure WebSockets and VPNs: For environments where TLS might not be suitable, using secure WebSockets or operating within a Virtual Private Network (VPN) can offer additional security layers.
- Certificate management: Ensuring proper management and regular updates of certificates used in TLS/SSL is crucial for maintaining encryption integrity.

Client Identifier (ClientID) Management

The structure and validation of ClientIDs are critical for defining access rights. A well-designed ClientID can be integrated into the topical structure, ensuring that clients only

access permitted topics. Key strategies include:

- Dynamic integration of ClientID in topic structure: This strategy allows for the dynamic allocation of rights based on the ClientID, providing a secure way to manage access at the topic level.
- Validation and persistence of ClientIDs: Ensuring that ClientIDs can be validated and are consistent across sessions enhances security. This also facilitates the use of persistent sessions, beneficial in unstable network conditions.
- Meta-information utilization: Using meta-information from ClientIDs for authorization purposes adds an additional layer of security by ensuring that only clients belonging to certain groups can access specific topics.

Securing Your MQTT Infrastructure for UNS

Securing MQTT infrastructure involves understanding network topology and implementing measures to prevent unauthorized access and system downtimes. Key strategies include:

- Firewall: Use firewalls to filter traffic, blocking unexpected or unnecessary traffic like UDP and ICMP packets, while allowing MQTT traffic on standard ports (1883 for TCP, 8883 for TLS).
- Load balancer: Employ load balancers to distribute MQTT traffic across multiple brokers, preventing overloading and enabling traffic throttling in high-traffic scenarios.
- Demilitarized zone (DMZ): Set up a DMZ for internet-facing services like MQTT brokers, with additional firewall protection to secure access to internal systems.

High Availability and Redundancy

Setting up MQTT broker clusters in each plant and a central cluster in the cloud can mitigate the risk of single points of failure. This approach ensures that even if one broker is compromised, others can take over, maintaining network integrity. Key strategies include:

- MQTT broker clusters: Implementing MQTT broker clusters in each plant and a central cluster in the cloud enhances resilience. In this setup, if one broker fails or is compromised, others in the cluster can continue to operate, minimizing downtime and potential data loss.
- Geographical distribution: Distributing these clusters across multiple availability zones can further reduce risks related to regional outages or disasters.
- Failover mechanisms: Implementing automated failover mechanisms ensures a seamless transition between brokers in case of failure.

Data Governance and Training

This approach ensures that everyone understands the system's structure and the importance of security, reducing the risk of inadvertent breaches. Key strategies include:

- Comprehensive data governance: This involves the entire architecture, ensuring all users know and comply with data security policies.
- Reskilling and training: Regular training programs for staff to understand the security implications and proper use of MQTT systems are vital. This helps in building a culture of security awareness across the organization.
- Decentralized data governance: Encouraging a decentralized approach, where every employee acts as a data steward, can enhance security by distributing responsibility.

Regular Updates and Monitoring

Keeping MQTT brokers and clients in your UNS updated and employing monitoring tools for anomaly detection is vital for maintaining a secure environment. Key strategies include:

- Automated patch management: Keeping MQTT brokers and clients updated with the latest security patches is crucial. Automating this process ensures the timely application of updates.
- Anomaly detection: Implementing advanced monitoring tools that can detect unusual network activity or access patterns helps in the early identification of potential security breaches.

Conclusion

The security of a UNS is a multifaceted endeavor requiring a comprehensive and proactive approach. As detailed in this article, employing strategies such as robust authentication and authorization, encryption and secure communication, and diligent ClientID management are foundational to securing UNS environments. Equally important are the infrastructural aspects, including effective firewall implementation, load balancing, and the establishment of demilitarized zones.

High availability and redundancy through MQTT broker clusters and geographical distribution ensure continuity and resilience in the face of potential breaches or system failures. Additionally, a strong emphasis on data governance and continuous staff training creates a culture of security awareness and readiness, which is crucial in mitigating risks associated with human error.

Ultimately, the security of UNS is not a one-time task but an ongoing process of adaptation and improvement. As technology evolves and new threats emerge, organizations must remain vigilant and responsive, continuously refining their security strategies to safeguard their operations and data effectively.

About HiveMQ

HiveMQ empowers businesses to transform with the most trusted MQTT platform. Designed to connect, communicate, and control IoT data under real-world stress, the HiveMQ MQTT Platform is the proven enterprise standard for Industry 4.0. Leading brands like Audi, BMW, Liberty Global, Mercedes-Benz, Siemens, and ZF choose HiveMQ to build smarter IIoT projects, modernize factories, and create better customer experiences.

Visit hivemq.com to learn more

HIVEMQ

www.hivemq.com