HighByte

Data Modeling Guidebook

How to leverage ISA-95

Modeling best practices and pitfalls

How to get started with modeling industrial data

Abstract

The digital transformation is providing industrial organizations with an unprecedented amount of visibility and predictive insights into their operations. Every day, factories and other industrial environments are adopting new, smart technologies that are delivering vast amounts of data they can use to optimize production, predict machine failures, and improve quality. However, connecting these machines to data storage platforms or enterprise systems isn't always seamless. It requires a standardized approach to defining and categorizing data, so everyone across the organization has a single source of truth and can make faster, more informed decisions.

This guidebook provides a comprehensive overview of data modeling and why it's essential for every industrial environment – no matter where the organization stands along its digital journey. In the following sections, readers will gain a better understanding of how data modeling works, what it looks like, how it works with existing standards (such as ISA-95), and tips on how to establish a data-modeling strategy.

Table of Contents

An Introduction to Data Models and Modeling

The data model forms the basis for standardizing data across a wide range of raw input data. An industrial DataOps solution like HighByte Intelligence Hub enables users to develop models that standardize and contextualize industrial data. In short, HighByte Intelligence Hub is a data hub with a modeling and transformation engine at its core.

Here are six commonly asked questions people have when they're considering a data-modeling solution.

01 What is a data model?

A data model describes a rich piece of information. The model consolidates information to fully address a specific use case. Use cases for industrial data can include supervisory monitoring of equipment or process lines, asset predictive maintenance, product or batch traceability, quality and productivity analysis, cost accounting, or simply maintaining information synchronization between multiple systems. The information can have many different attributes, some containing real-time, raw operational data and others that actually define that data. The latter provides context, which could be a source description, unit of measure, min and max ranges, and other types of information that—when pulled together—define a piece of information that corresponds to a real "thing" like an asset, process, system, or role.

While data models are well-known entities in the industrial automation industry, they go by many names. Depending on a user's experience or function, they may be accustomed to different naming conventions. For instance, a controls engineer modeling complex data sets within a PLC might refer to the data model as a "user-defined type." Others associate data models with the library of built-in complex tags that are provided by default based on the PLC vendor's implementation. Still, others may think of data models as simply the structure—such as names, data points, and data types, and whether these properties are required.

The industry also has come together to create standardized data models or data sets, like ISA-95 (discussed in the next section), MTConnect, and companion specs to OPC UA, which are sometimes specific to a particular vertical industry. These models define nomenclature, how data should be represented, and the model's structure (how it is laid out).

However, there are still many data models in use that are vendor specific (sometimes even device specific) and, therefore, not standardized across the industry. Information technology (IT) systems and cloud applications also have their own requirements on how data should be modeled, received, and stored. Among operational technology (OT) and IT devices, systems, and applications, there is a lot of data model diversity and little standardization in real-world practice.

02 Why is data modeling important?

Data modeling is important because models standardize information, enable interoperability, show intent, determine trust, and ensure proper data governance.

To expand on these ideas, data modeling enables:

STANDARDIZATION

How data is categorized and pulled together for additional meaning.

INTEROPERABILITY

The ability for multiple users across different job functions to look at data and quickly understand its source, structure, and what the model represents (like a pump or a production line). This context and metadata are what makes modeling so important.

DATA INTEGRITY

Ensures the intent of the data is clear, including its value, what it represents, if it’s in an acceptable range, and whether it can be trusted.

DATA GOVERNANCE

Dictates how information should be shared across business units and mandates data uniformity. It also ensures only the appropriate systems and users who need access to that information receive the data and understand it. By modeling data with an abstraction layer dedicated to merging, modeling, and securely sharing data, we help ensure proper data governance.

03 Why do we need a dedicated layer for data modeling?

A dedicated abstraction layer, also referred to as the DataOps layer, is essential because not every application conforms to one standard. Industry standards exist but they are finite. As such, vendors continue to create their own schemas to model rich information in the context of their application. Fortunately, vendors typically provide some level of an API to push and pull data from the application in the expected format that a dedicated data modeling layer can then leverage.

By orchestrating these integrations within a dedicated layer, we can begin to genericize data modeling, such that a user can work in a *single environment* to model any number of things. That layer then becomes responsible for transforming data into the specific data-modeling schemas for all consuming applications. This is game changing for users who need to collect, merge, transform, and share information with many applications that live on-premises and in the cloud. Users can build out or deploy new applications over time and take advantage of previous work. By managing data modeling in a centralized location, users can add, delete, and edit parameters for connected applications without breaking existing integrations.

Other key benefits of an abstraction layer include:

VISIBILITY

Automation engineers often know they have hardware and software on the plant floor that are producing and collecting raw data, but they don't know who is connecting to them and what data is being shared. A centralized location allows OT to easily view where and how data flows in and out of the plant floor. They know who is producing the raw data, who is consuming the information, and how changes to the DataOps layer will impact the rest of the enterprise. A dedicated layer adds resiliency and flexibility to the vast ecosystem of technology found in most manufacturing facilities.

REDUCED DATA PREPARATION AND INTEGRATION TIME

Information can be automatically propagated to any vendor's application without touching each application individually. It's time consuming to model data in many different applications as opposed to modeling data just once.

LESS DOWNTIME

A DataOps layer provides passive connectivity—meaning, users won't need to schedule downtime or rewire integrations to establish communication with the solution. An industrial DataOps solution, like HighByte Intelligence Hub, can passively drop in, make connections to existing data sources, pull data, transform it, add context to it, and then push out real-time modeled information to running applications using their respective APIs.

Other key benefits of an abstraction layer include:

FEWER ERRORS

The DataOps layer transforms raw data before making it available to all consuming applications, so there is less chance of errors occurring upstream—like attaching the wrong units of measure to a data point. DataOps reduces the opportunity for human error by providing a central location to manage conversions and transformations. If there is an error, it's detected quickly and easily fixed without troubleshooting each application or mining custom code.

CONSISTENCY

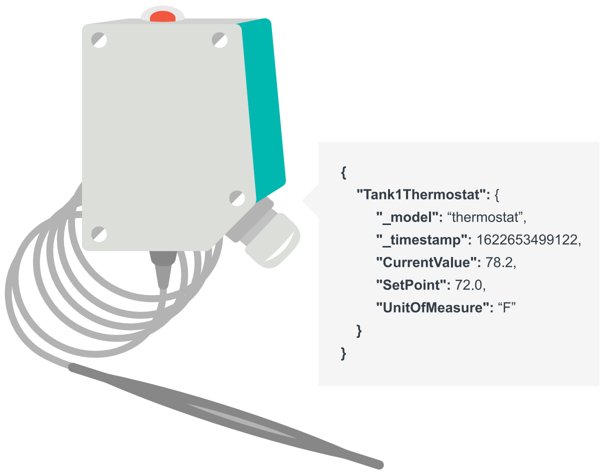
For data that includes time characteristics, the DataOps layer helps ensure time stamps are consistent and accurate across all applications. When applications are allowed to collect data directly and independently, they may collect different time samples depending on the rate-of-change velocity of the underlying data. Instead, by using a DataOps layer, users ensure all applications receive the same time sample and are in synch with one another, down to millisecond resolution.

SECURITY

When industrial companies manage data modeling and integration in a dedicated DataOps solution, they bolster their defense-in-depth strategy. The modeling environment enables only authorized individuals to determine which applications should receive data and exactly what data they should receive. Consuming applications no longer have unfettered access to raw data sources. The DataOps solution abstracts away this direct connection. And rather than burying integrations in custom code, they are visible to authorized users and help protect potentially critical infrastructure.

04 What does a data model look like?

A data model is not—and should not—be complicated. At its most basic definition, a data model is one-to-many name-value pairs. Data models are created as logical collections of these name-value pairs that are related in some way and—when put together—become a valuable and useful information object.



```
{
  "Tank1Thermostat": {
     "_model": “thermostat”,
     "_timestamp": 1622653499122,
     "CurrentValue": 78.2,
     "SetPoint": 72.0,
     "UnitOfMeasure": “F”
  }
}
```

For example, an engineer might create a data model that represents a thermostat. The first attribute is a current value. The second attribute is a set point value. The third attribute is a unit value. The model clearly articulates how a thermostat should be represented for the enterprise. In this example, every thermostat will have a name, current value (a floating-point value), set point, and unit of measure (a static character indicating degrees Fahrenheit or degrees Celsius).

A thermostat is obviously a simple thing to model. But this same concept applies to even the most complex process or piece of equipment. The model is distilled down to its primitive, most important data points. Then contextual attributes can be added to the model to describe what the data points are and what they should be, so the information becomes self-describing to anyone viewing it. The top-level data model and any sub-models define the structure of the data payloads which will be transmitted to the target system.

05 What's the difference between a model and instance?

In HighByte Intelligence Hub, users can build models and instances, which are the blueprints of how they want to structure data about a generic asset, product, process, or system.

Let's say, for example, that a manufacturer has 10 pumps, and wants to model all of these assets in a certain way to standardize the data they produce. The data attributes these pumps have in common will form the model. The variation between pumps will result in a unique instance. So, in this example, there is one model and 10 instances that represent those 10 real-world pumps.

After the model is built, the instances are created and populated with the appropriate connections and data-access references needed to collect the model-defined data from the appropriate pump and populate the real-time instance that can be shared with a consuming application.

Users may have many instances of a single model, but building the instances doesn't need to be time consuming. HighByte has designed the Intelligence Hub with scalability and user experience in mind. It's easy to clone instances, reuse as much as possible, and then only modify specific attributes as necessary.

06 What's the most common mistake when users first start modeling data?

There are a few common pitfalls that impact users when they first begin modeling data in an abstraction layer like HighByte Intelligence Hub.

The most common mistake is adding instance-specific attributes inside the model. So, for example, instead of creating a property in the model named "Current value," a user creates a property named "Thermostat 1 current value." Now the user has tied an instance-specific value to a model, which was intended to be the generalized blueprint. This won't scale well and can't be reused. HighByte advises its customers to keep uniqueness out of the model, and instead reserve unique properties for the instances they define from the generic model. How users populate the attributes defined by the model will add to the uniqueness of each instance.

The second most common pitfall is starting with models that are too complex. Many manufacturing-specific data tools require companies to collect every piece of data they might need and wire it correctly the first time, knowing that making changes to the system down the road would be extremely difficult. HighByte built the Intelligence Hub with this problem in mind. Because the software passively collects and shares data without interrupting operations, users can start small and iterate over time with very little pain. HighByte advises customers to begin with a specific, critical problem or use case and identify the data required to solve for this problem. Using HighByte Intelligence Hub, an engineer can then connect the sources and targets, build the model and instances, and establish flows to get the modeled information to its destination.

It’s also important to make the most critical attributes of the model required fields. Such that, if a user builds an instance from the model, the user must populate these fields to save the instance. The models are adaptable in HighByte Intelligence Hub, so over time users can make changes as their needs evolve by requiring additional attributes, making them optional, or removing them altogether.

Data modeling doesn’t need to be complex. Start small. Effective models distill data sets down to their simplest form so they can easily be reused, helping manufacturers achieve standardization at scale.

Leveraging ISA-95

The previous chapter mentioned that there have been industry efforts to standardize data models. While there are many standards available to guide companies in modeling and structuring their data, ISA-95 is the most commonly recognized standard around the world.

Introduction

ISA-95 has been implemented globally in many manufacturing plants and is the guide for many off-the-shelf and bespoke manufacturing execution systems (MES). The ISA-95 specification defines itself as “the interface content between manufacturing operations and control functions and other enterprise functions. The interfaces considered are the interfaces between Levels 3 and 4 of the hierarchical model defined by this standard.”

ISA-95 provides a number of helpful models that are great starting points when looking to implement data integrations that link MES, enterprise systems, IIoT, data lakes, and analytics. It also eases the implementation of a unified namespace (UNS) for enterprise data integration. A UNS is a consolidated, abstracted structure by which all business applications are able to consume real-time industrial data in a consistent manner.

The specification defines a hierarchal model for systems, detailed information models, and a data flow model for manufacturing operations management (MOM).

- ✓ **First, this article will explain these core elements of the ISA-95 specification.**
- ✓ **Then, it will demonstrate how the ISA-95 specification can be applied within HighByte Intelligence Hub.**

The ISA-95 Specification

HIERARCHAL MODEL

The hierarchal model in the ISA-95 specification provides a convenient way of thinking about information organization and structuring information at different levels of the corporation. In today's environment with data lakes, UNS, and systems that aggregate information from multiple cells, lines, areas, sites and even enterprises, this organization is very helpful when organizing and annotating information. Manufacturers will find data and systems aligning to different levels depending on their unique production environment and circumstance.

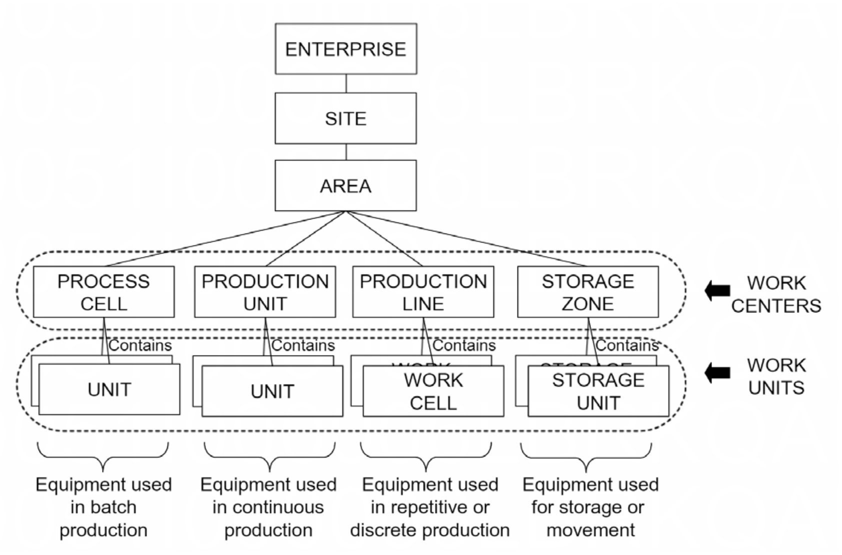

ENTERPRISE
SITE
AREA
PROCESS CELL
PRODUCTION UNIT
PRODUCTION LINE
STORAGE ZONE
WORK CENTERS
Contains
Contains
Contains
Contains
UNIT
UNIT
WORK CELL
STORAGE UNIT
WORK UNITS
Equipment used in batch production
Equipment used in continuous production
Equipment used in repetitive or discrete production
Equipment used for storage or movement

***Figure 1**: Example of defined types of work centers and work units as defined by the ISA-95 specification.*

Image courtesy of the International Society of Automation.

INFORMATION MODELS

The models within the ISA-95 specification can get very complex and detailed. These models span multiple structures including personnel, equipment, physical assets, and material. They also span production, maintenance, quality, and inventory disciplines. The ISA-95 specification defines the inter-relationships.

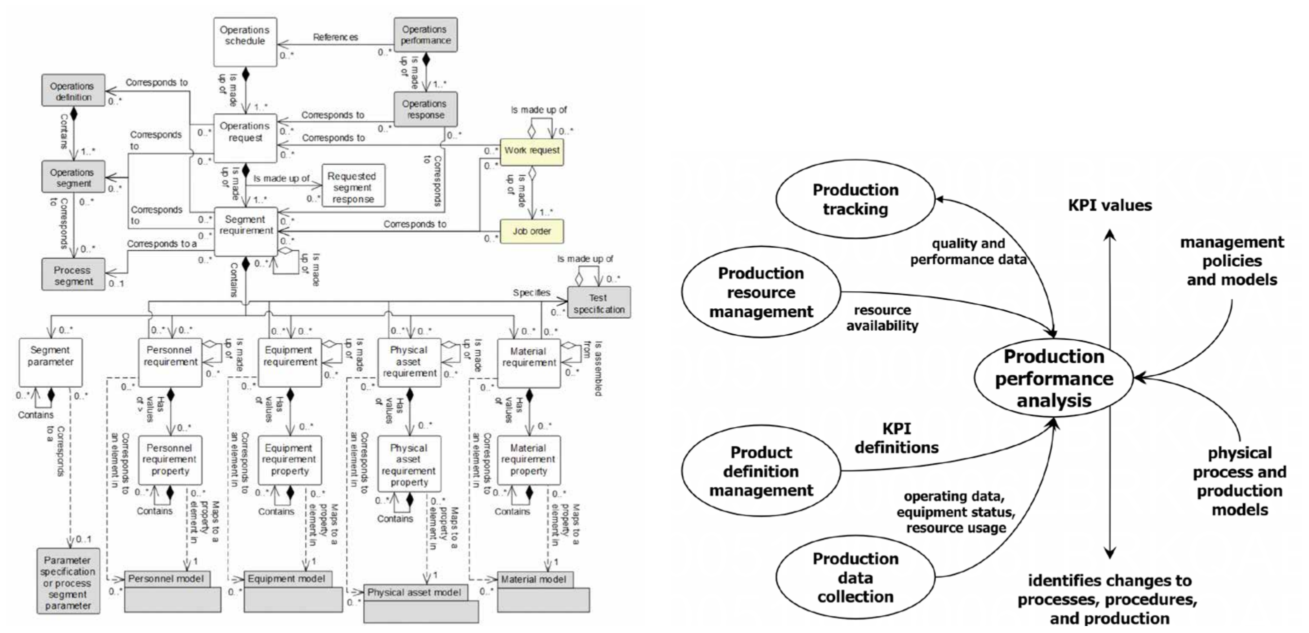

Operations schedule
References
Operations performance
Operations definition
Corresponds to
Operations request
Operations response
Is made up of
Work request
Operations segment
Requested segment response
Job order
Segment requirement
Process segment
Corresponds to a
Contains
Specifies
Test specification
Segment parameter
Personnel requirement
Equipment requirement
Physical asset requirement
Material requirement
Has values of
Personnel requirement property
Equipment requirement property
Physical asset requirement property
Material requirement property
Is assembled from
Maps to a property element in
Corresponds to an element in
Parameter specification or process segment parameter
Personnel model
Equipment model
Physical asset model
Material model
Production tracking
quality and performance data
KPI values
management policies and models
Production resource management
resource availability
Production performance analysis
KPI definitions
Product definition management
physical process and production models
operating data, equipment status, resource usage
Production data collection
identifies changes to processes, procedures, and production

Figure 2: Operations performance model (left) and production performance analysis activity model interfaces (right) as defined by the ISA-95 specification.

Image courtesy of the International Society of Automation.

DATA FLOWS

The ISA-95 specification goes into very deep definition around the flow of data from different functional models. These functional models could be in a single system or they can be in separate systems depending on the corporation and manufacturing environment. When integrating with separate systems, it's important to identify the use case, then the source and target system's data requirements, and then define the execution or triggering event. These data flows can be modeled as discrete connections between systems, handled through a data hub, or handled through a UNS.

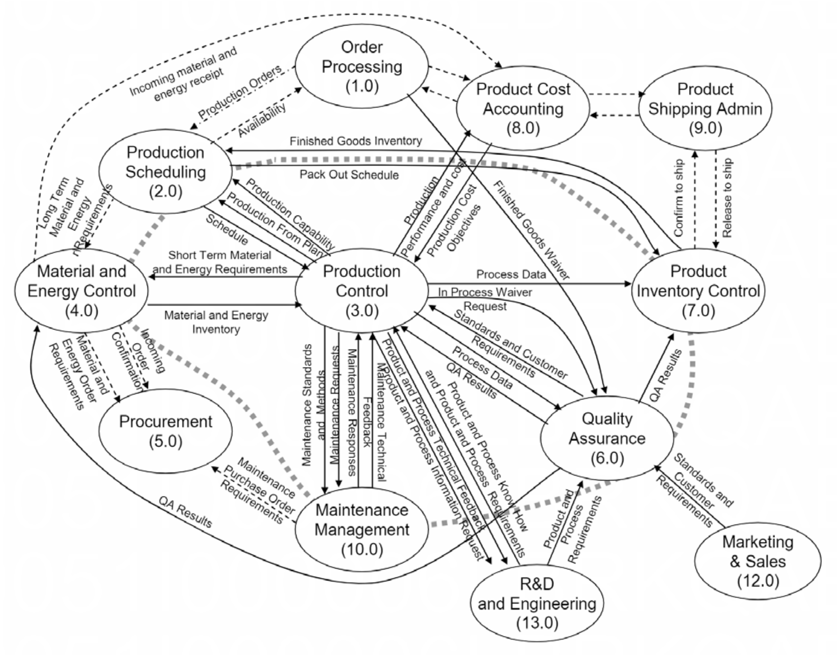

Order Processing (1.0)
Production Scheduling (2.0)
Production Control (3.0)
Material and Energy Control (4.0)
Procurement (5.0)
Quality Assurance (6.0)
Product Inventory Control (7.0)
Product Cost Accounting (8.0)
Product Shipping Admin (9.0)
Maintenance Management (10.0)
Marketing & Sales (12.0)
R&D and Engineering (13.0)
Incoming material and energy receipt
Production Orders
Availability
Finished Goods Inventory
Pack Out Schedule
Long Term Material and Energy Requirements
Production Capability
Production From Plan
Schedule
Short Term Material and Energy Requirements
Material and Energy Inventory
Production Performance and cost
Production Cost Objectives
Finished Goods Waiver
Confirm to ship
Release to ship
Process Data
In Process Waiver Request
Standards and Customer Requirements
Process Data
QA Results
QA Results
Material and Energy Order Requirements
Incoming Order Confirmation
Maintenance Standards and Methods
Maintenance Requests
Maintenance Responses
Maintenance Technical Feedback
Product and Process Information Request
Product and Process Technical Feedback
and Product and Process Requirements
Product and Process Know How
Product and Process Requirements
Standards and Customer Requirements
Maintenance Purchase Order Requirements
QA Results

Figure 3: *A functional model as defined by the ISA-95 specification.*

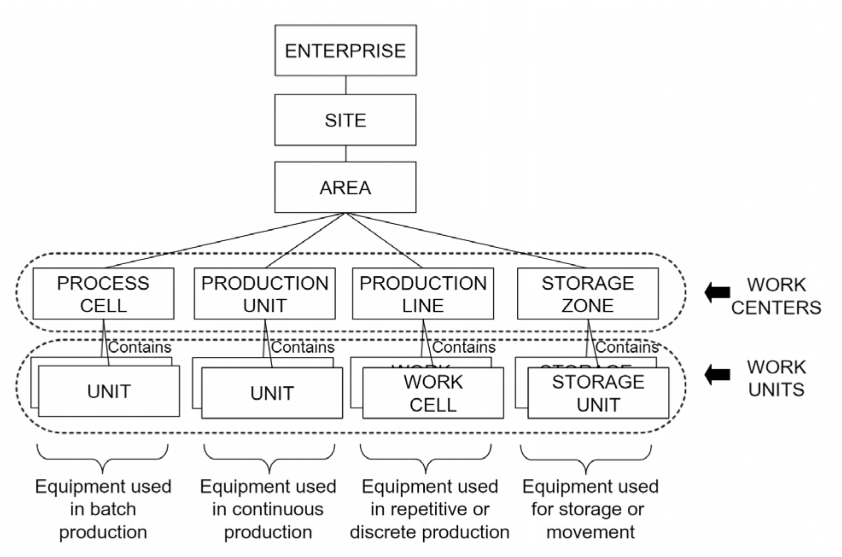
Image courtesy of the International Society of Automation.

HighByte Intelligence Hub & ISA-95

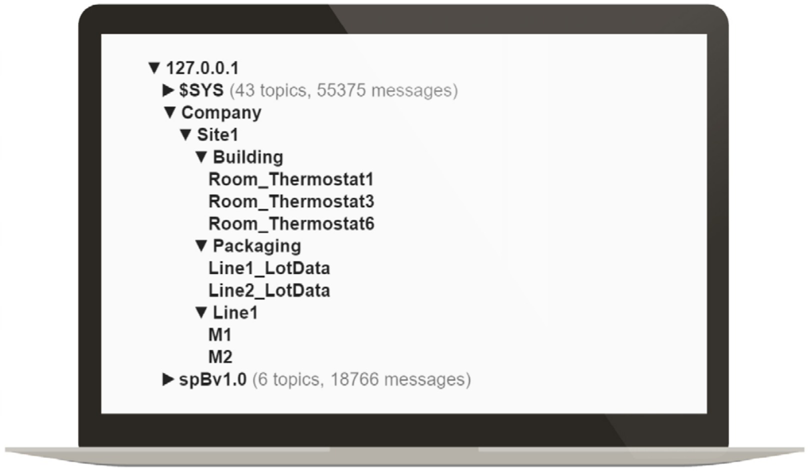
HighByte Intelligence Hub serves as the integration solution between systems at the same level or different levels in the ISA-95 stack. It also can populate a unified namespace structured to the ISA-95 hierarchy and can be used to structure a data lake, data warehouse, or database using the ISA-95 models.

DATA ORGANIZATION

The hierarchal model defined in ISA-95 is applicable for organizing data payloads in a UNS and organizing models and instances within HighByte Intelligence Hub. Within the Intelligence Hub, users can set up MQTT outputs with the Enterprise, Site, Area, Line, Cell, or Machine topic structure, so they can organize payloads of data at the appropriate layer in the broker. In addition, this same structure provides a convenient way to organize the storage of models and instances.



ENTERPRISE
SITE
AREA
PROCESS CELL
PRODUCTION UNIT
PRODUCTION LINE
STORAGE ZONE
WORK CENTERS
Contains
UNIT
UNIT
WORK CELL
STORAGE UNIT
WORK UNITS
Equipment used in batch production
Equipment used in continuous production
Equipment used in repetitive or discrete production
Equipment used for storage or movement

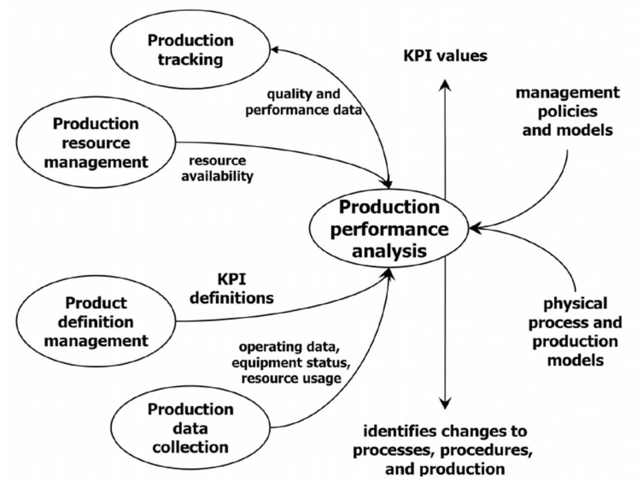


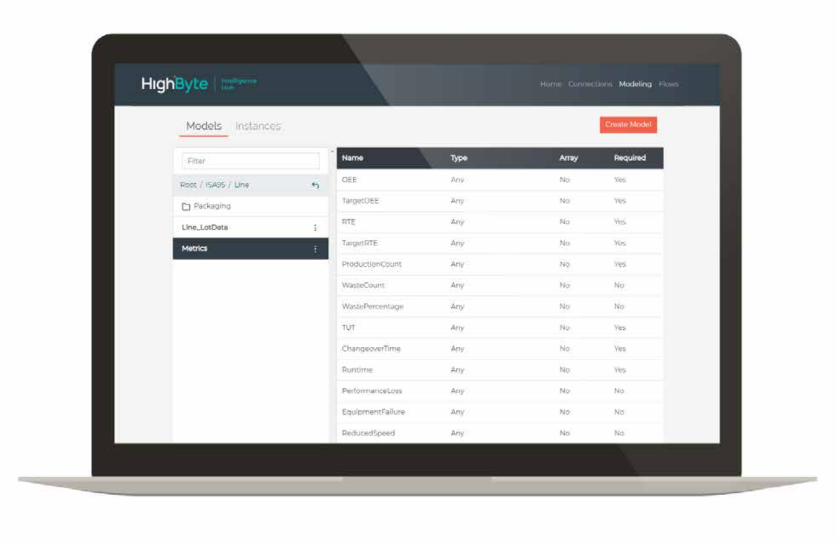
```
▼ 127.0.0.1
  ▶ $SYS (43 topics, 55375 messages)
  ▼ Company
    ▼ Site1
      ▼ Building
         Room_Thermostat1
         Room_Thermostat3
         Room_Thermostat6
      ▼ Packaging
         Line1_LotData
         Line2_LotData
      ▼ Line1
         M1
         M2
  ▶ spBv1.0 (6 topics, 18766 messages)
```

Figure 4: The image on the right illustrates payload organization in an MQTT output from HighByte Intelligence Hub, demonstrating how the hierarchal model on the left in the ISA-95 specification (see Figure 1) can be applied to HighByte Intelligence Hub to organize and structure information at different levels of the corporation.

MODELS AND INSTANCES

There are many "models" or data structures defined in the ISA-95 specification for storing and working with data. These complex models are required for the MES/MOM system to operate and maintain data normalization and integrity but are not necessarily needed when interfacing data. When interfacing data, users need to define the required information payload for the target system or analytic. In HighByte Intelligence Hub, a model comprises multiple attributes and other models. This allows logical models to be built for specific integrations by reusing base components.


Production tracking
Production resource management
Product definition management
Production data collection
Production performance analysis
quality and performance data
resource availability
KPI definitions
operating data, equipment status, resource usage
KPI values
management policies and models
physical process and production models
identifies changes to processes, procedures, and production



Name	Type	Array	Required
OEE	Any	No	Yes
TargetOEE	Any	No	Yes
RTE	Any	No	Yes
TargetRTE	Any	No	Yes
ProductionCount	Any	No	Yes
WasteCount	Any	No	No
WastePercentage	Any	No	No
TUT	Any	No	Yes
ChangeoverTime	Any	No	Yes
Runtime	Any	No	Yes
PerformanceLoss	Any	No	No
EquipmentFailure	Any	No	No
ReducedSpeed	Any	No	No

Figure 5: The image on the right demonstrates the modeling UI in HighByte Intelligence Hub, demonstrating how the information models in the ISA-95 specification (see Figure 2) can be applied to HighByte Intelligence Hub to model production performance.

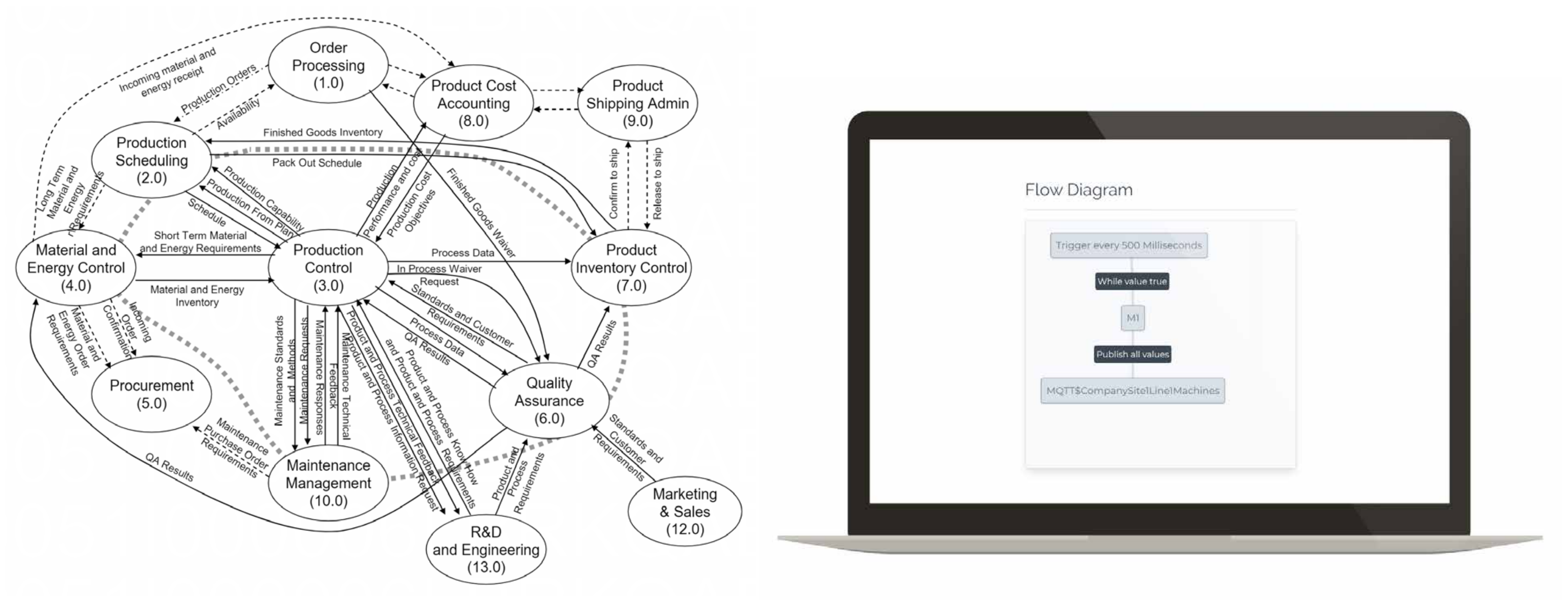
ISA-95 Model Scope

The ISA-95 model does not define specific manufacturing asset data, which is required for many systems. For instance, ISA-95 doesn't specify the specific attributes of a pump, i.e. enabled bit, pressure, flow, manufacturer, model number, serial number, installation data, etc. In many cases, industry groups made up of customers and manufacturers within a specific industry have developed asset models and metric models for use within the applications.

The ISA-95 specification also does not specify asset-level data or analytical data. Traceability and analytical data often span multiple levels of the ISA-95 hierarchy and multiple models within a level. It's not uncommon for manufacturers to merge sensor data with PLC, SCADA, MES, ERP, and other systems that live at different levels.

DATA FLOW

HighByte Intelligence Hub data flows can map directly to the ISA-95 data flows. Any flows from system to system can be implemented within HighByte Intelligence Hub as defined in the specification. Organizations also can easily implement additional data flows that leverage the same information but are outside the scope of ISA-95.



Order Processing (1.0)
Production Scheduling (2.0)
Production Control (3.0)
Material and Energy Control (4.0)
Procurement (5.0)
Quality Assurance (6.0)
Product Inventory Control (7.0)
Product Cost Accounting (8.0)
Product Shipping Admin (9.0)
Maintenance Management (10.0)
Marketing & Sales (12.0)
R&D and Engineering (13.0)
Incoming material and energy receipt
Production Orders
Availability
Finished Goods Inventory
Pack Out Schedule
Long Term Material and Energy Requirements
Production Capability
Production From Plan
Schedule
Short Term Material and Energy Requirements
Material and Energy Inventory
Production Performance and cost
Production Cost Objectives
Finished Goods Waiver
Process Data
In Process Waiver Request
Standards and Customer Requirements
Process Data
QA Results
Confirm to ship
Release to ship
Material and Energy Order Requirements
Order Confirmation
Incoming
Maintenance Standards and Methods
Maintenance Requests
Maintenance Responses
Feedback
Maintenance Technical
Product and Process Information Request
Product and Process Technical Feedback
Product and Process Know How
Product and Process Requirements
Maintenance Purchase Order Requirements
QA Results
Product and Process Requirements
Standards and Customer Requirements
QA Results
Flow Diagram
Trigger every 500 Milliseconds
While value true
M1
Publish all values
MQTT$CompanySite1Line1Machines

Figure 6: The image on the right demonstrates the flow UI in HighByte Intelligence Hub, demonstrating how the functional models in the ISA-95 specification (see Figure 3) can be applied to HighByte Intelligence Hub to flow information between systems.

Tips for Applying ISA-95

When applying a standard and planning your system integration, it’s important to remember a couple things:

01 KEEP PROJECTS AS SIMPLE AS POSSIBLE

It’s easy to turn a data-modeling project into an academic and bureaucratic process trying to develop the perfect model for everyone. In the end, these projects typically take far longer than they need to and don’t result in better outcomes. Most modern systems are flexible and agile and can be changed over time. The priority should be on getting the process started and working with the data while being flexible to make adjustments as needed. Modern analytics requires an agile environment where the data collected can change based on the results of the analysis.

02 FOCUS ON THE GOAL

New use case, new system, new integration architecture. When integrating systems, it’s important to focus on the use case, the target system, and the data. Then, work backward to the required information, the possible sources of this data, and the transformations that will be required to convert raw data into information.

Conclusion

The ISA-95 specification provides many useful hierarchies, models, and flows that are applicable when integrating industrial data. These models and structures serve as a starting point and guide when defining a DataOps solution. Even so, a company may need to deviate or extend from ISA-95 for its specific application, use case, and system architecture. The data also will require more details than are specified in ISA-95. These additional details will come from asset-specific standards, the integrated systems, and specific use cases the company is looking to solve.

Manufacturers can easily map the components of ISA-95 to HighByte Intelligence Hub and add any additional customization that's required. By using the Intelligence Hub, the OT team will be able to see and modify the data hierarchy, structures, and flows in an application that promotes re-use, agility, and the flexibility to easily make changes over time.

Best Practices and Pitfalls

Data-modeling projects may seem daunting at first glance. There's no question that manufacturers must take a number of factors into consideration to ensure their data-modeling project is a success. The following best practices serve as a guideline for commonly encountered challenges and best practices to overcome these hurdles.

FOCUS ON REQUIREMENTS OF END SYSTEMS AND USE CASES

Avoid model purgatory. In many cases, users who get stuck in modeling start by creating a logical view of a machine (it has a single press, two motors, a pump, etc.) and then they start working backward (it's in Area 2, at the Kentucky plant). They create this all as hierarchy.

Instead, it's best to start by asking, *What does the end application need?* Maybe that's an application like Maximo and the user needs to create a work order. *What's the minimal information needed from production to create that?* Maybe it's a Microsoft Power BI dashboard that leadership needs. *What's the minimal information they need to see in the dashboard?* The user can start by creating models around this minimal information and then add to them in the future.

If a second application comes up that needs a similar model, the user can leverage the existing model or create a new model specific to this application. This approach is better than "modeling the world" because, in this case, there is a 99% chance the end applications couldn't consume the world model anyway, so the user would need to create new models that pick off parts of the world model.

PLAN ON MODELS CHANGING

Creating models is similar to defining a class in programming. A programmer designs the class with an end application in mind, knowing that over time the requirements will evolve and mature and changes will be necessary. Accept this going into data modeling. The best defense against changing models is to model the bare minimum for the end application. It's easier to add to a model then it is to remove or change existing attributes.

MINIMIZE MODEL HIERARCHY WHENEVER POSSIBLE

Hierarchy scales complexity. As the model hierarchy grows, it becomes more difficult to manage in the end applications. Many applications don't support hierarchy, so users are required to flatten data models anyway. A simple example of this is site information. Rather than create a "Site" model, with a single name attribute, it's better to put a "Site" attribute in the machine model. This way, organizations avoid hierarchy, and the machine model is self-contained. If sites have three or four other properties, like address, employee count, etc., then it may be worth creating the hierarchy where a site contains 1-N machines, but it's better to avoid this when possible. Over time, as data hierarchy and modeling are adopted by more solutions, hierarchy will become easier to manage. But today, excessive hierarchy will make integrations harder.

INCLUDE LOCATION/METADATA IN THE MODEL ITSELF

Similar to the guidance above, a best practice is to make models as self-contained as possible. Think of it like storing a model in a database. Whenever possible, store all the information in a single table. This way, users can query one table/topic/source and get all the information the end application needs. In SQL it's easy to split up models. As an example, you might create separate tables for site and machine information, and do a table JOIN to get both sets of information. There is no concept of a JOIN in a UNS or MQTT broker and writing code to listen on multiple topics and pull the information together is messy and unreliable. It's best to load models up (using hierarchy or not) with all the information required in the end application.

VERSION THE MODELS

Models will change, and without versioning—especially as the UNS evolves—users will run into problems. For example, let's say there's a Microsoft Azure function in the cloud that subscribes to press machines in a UNS, parses the data, and feeds it off to a machine learning (ML) algorithm. The data science group requests a change. They need to receive "runtime" in milliseconds instead of seconds. There are two ways to do this:

1. Create a second attribute in the model called "runtimeMS," make it non-required, and fill this in with machine data as the organization rolls out the new model. The Azure function can check if runtimeMS is in the data, and if it is, assume it's the new model and route it to the appropriate ML function. This works, but it's a bit of a hack that only works when you add to a model.

2. Change the runtimeS attribute to runtimeMS and bump the model version from 1 to 2. In this case, the Azure function first checks the model version, and if it's 2, it runs it through the new ML function. This is a more robust solution that works with any change.

AVOID DUPLICATING MODELS IN DIFFERENT SYSTEMS

Where possible, use structured data coming from the sub-system. If the source system is producing modeled data (JSON, OPC UA, SQL, etc.), try to leverage those underlying models as much as possible. As a last resort, break the model up into its attributes and then reconstruct it.

KEEP DATA TYPES SIMPLE AND UNIFORM

The factory has a lot of data types (bytes, bits, floats, doubles, hex, etc.). This is because historically during the serial days we were concerned about bits and bytes. With modern Ethernet and fiber, this is less of a concern. The end result is users often need to map very specific data (ex. Int8) to very generic data like a numeric type in JSON. As a general rule, specific type information is lost as data moves up to cloud systems. Plan for this. To mitigate complexity, keep type information simple. When possible, treat everything like ints or floats.

IF USING MQTT, THINK ABOUT TOPIC NAMESPACE AND HOW MODELS FIT

Topics in MQTT (either standard or Sparkplug) contain metadata. For example, it's possible to publish machine1 data to a topic that is broken into an ISA-95 hierarchy (ex. Site1/Area2/Cell3/Machine1). In this scenario, the site, area, and cell names are all metadata. In general, it's best to include this metadata in the model so it's available in the payload as well. This way, the end application doesn't need to maintain the context of the topic path and can get all the context from the payload itself.

LIMIT THE NUMBER OF MODEL "INSTANCES"

A plant with 10 of the same machines may have production information from an MES stored in a database, each with a unique machine ID. They also have process information from an OPC UA server. In the general case, you'll create a model for the machine, and then create 10 instances of the model, one for each machine. This is required if the data mappings (in this case OPC UA tag addresses) are unique for each machine. The user needs some place for this uniqueness, and a good place for that is the instance. However, if instead the data is only coming from SQL, and each row is a single machine, the user doesn't need 10 instances. They just need a single instance that queries the rows, packages/manipulates the data (maybe changes a column name to something more human readable), and sends it to the UNS. In this case, 10 instances would be overhead.

USE A SINGLE TIMESTAMP

Factories are all about time-series data, and every data change of every tag has a timestamp. This is great in SCADA/HMI and historian applications. But anything outside of this represents a challenge. Modeled data isn't time-series data. IT/cloud systems want a snapshot of the machine at time X with the value for all of its attributes. Machine learning software like Tensorflow and others don't want to get a collection of tag value changes at various times and be required to stitch these together to find out what the values were at time X. In fact, they don't even want to deal with holes in the data where the application didn't send a value for attribute Y because it didn't change. Instead, they want consistent data that shows the state of the machine at time X, X+100ms, X+200ms, etc. So, specifically for ML/artificial intelligence (AI) applications, users must make sure they're delivering the entire model with a single timestamp. Typically, that timestamp is UTC and in epoch (milliseconds since midnight, January 1970).

DON'T PASS UP QUALITY

Traditional OT data from OPC contains a value, quality, and timestamp for each tag. Historically quality was meaningful when communicating over serial (RS-232/485). The device could send a message that got "distorted" and the OPC server would report the quality as bad or unknown. Or more commonly, a device would go offline and the OPC server reported bad quality.

Traditional IT systems don't care about quality. They assume any data they get is "good".

When possible, don't include data quality in modeling. HighByte Intelligence Hub by default won't send data for an instance if one of its required attributes has bad quality data.

Instead, use quality as a way to notify people and applications that connectivity is lost. For example, you might trigger a flow to notify someone via text message by posting an alert to Twilio.

How to Get Started

Data-modeling strategies are only useful if manufacturers have a strategic plan to make use of the staggering volumes of data they receive on an ongoing basis.

Unfortunately, many manufacturers are drowning in data and struggling to make it useful.

A modern industrial facility can easily produce one terabyte of data each day. With a wave of new technologies for AI and ML—on top of real-time dashboards and augmented reality—manufacturers should realize significant productivity gains. In this connected world, unplanned asset and production line maintenance should be a thing of the past. But that's not the case for organizations that are simply collecting raw industrial data. They must make the data "fit for purpose" to extract its true value. Also, the tools they use to make the data fit for purpose must operate at the scale of an industrial facility.

By following these key steps, manufacturers and other industrial companies can extract the most value from their data models:

01 START WITH THE USE CASE

IT and OT projects should all start with clear use cases and business goals. For many manufacturing companies, projects may focus on machine maintenance, process improvements, and/or product analysis to improve quality or traceability. As part of the use case, company stakeholders should identify the scope of the project and the applicable data that will be required. Make sure the right cross-functional stakeholders are in the room from the beginning of the project, and that all stakeholders agree to prioritize the project and can reach consensus on the project goals.

02 IDENTIFY THE TARGET SYSTEMS

With the use cases and business goals identified, the next step requires identifying the target applications that will be used to accomplish these goals. Characterize the target application by asking these questions:

- Where is this target application located: at the edge, on-premises, in a data center, in the cloud, etc.?
- How can this application receive data: MQTT, OPC UA, REST, database load, etc.?
- What information is needed for this application?
- How frequently should the data be updated and what causes the update?

Document responses and then move on to the next step.

03 IDENTIFY THE DATA SOURCES

Industrial data is an important component for addressing industrial and business use cases. However, there are some major challenges with accessing this data and converting it into useful information.

Volume. The typical modern factory has hundreds to thousands of pieces of machinery and equipment that are constantly creating data. This data is generally aggregated within PLCs, machine controllers, or DCS systems within the automation layer, though newer approaches may also include smart sensors and smart actuators that feed data directly into the software layer.

Correlation. Automation data was primarily put in place to manage, optimize, and control the process. The data is correlated for process control and is not correlated for asset maintenance or product quality or traceability purposes.

Context. Data structures on PLCs and machine controllers have minimal descriptive information—if any. In many cases, data points are referenced with cryptic data point naming schemes or references to memory locations.

Standardization. The automation in a factory evolves over time with machinery and equipment sourced from a wide variety of hardware vendors. This hardware was likely programmed and defined by the vendor. This has resulted in unique data models created for each piece of machinery and a lack of standards across the factory and company in all but the very largest and most sophisticated manufacturers.

Organizations can better understand the specific challenges they will need to overcome for their project by documenting their data sources. Characterize the data available to meet the target system's needs by asking these questions:

- What data is available?
- Where is it located: PLCs, machine controllers, databases, etc.?
- Is it real-time data or informational data (metadata)?
- Is the data currently available in the right format or will it need to be derived?

04 SELECT THE INTEGRATION ARCHITECTURE

Integration architectures fall in two camps: direct API connections (application-to-application) or integration hubs (DataOps solutions).

Direct API connections work well if you only have two applications that need to be integrated, the data does not need to be curated or prepared for the receiving application, and the source systems are very static. This is typically successful in environments where the manufacturing company has a single SCADA or MES solution that houses all of the information, and there is no need for additional applications to get access to the data.

Direct API connections don't work well when industrial data is needed in multiple applications like SCADA, MES, ERP, IIoT platforms, analytics, QMS, AMS, cyber-threat monitoring systems, various custom databases, dashboards, or spreadsheet applications. Direct API connections also don't work well when there are many data transformations that must occur to prepare the data for the consuming system. These transformations can easily be performed in Python, C# or any other programming language, but they are then "invisible" and hard to maintain. Finally, direct API connections don't work well when data structures are frequently changing. This happens when the factory equipment or the programs running on this equipment are frequently changed. For example, a manufacturer may have short-run batches that require loading new programs on the PLC. The products the manufacturer produces may evolve and require changes to the automation. The manufacturer may change the automation system to improve efficiency. Or, the company may replace the equipment due to age and performance. Using the API approach buries the integrations in code. Stakeholders may not even be aware of integrated systems until long after the equipment has been replaced or changes have been made, resulting in undetected bad or missing data for weeks or even months.

An alternative to direct API connections is a DataOps integration hub.

05 ESTABLISH SECURE CONNECTIONS

Now that the project plan is in place, begin system integration by establishing secure connections to the source and target systems. Users must understand the protocols they will be working with and the security risks and benefits they provide.

Many systems support open protocols to define the connection and communication. Typical open protocols include OPC UA, MQTT, REST, ODBC, and AMQP—among others. There are also many closed protocols and vendor-defined APIs for which the vendor of the application publishes the API protocol documentation. Organizations should ask themselves: Does the protocol support secure connections and how are these connections created?

Some protocols and systems support certificates exchanged by the applications. Other protocols support usernames and passwords or tokens manually entered into the connecting system or through third-party validation. In addition to user security, some protocols support encrypted data packets so if there is a "man in the middle" attack they cannot read the data being passed. Some protocols support data authentication—so even if the data is viewed by a third party, it can't be changed.

Security is not just about usernames, passwords, encryption, and authentication but also about integration architecture. Protocols like MQTT require only outbound openings in firewalls, which security teams prefer because hackers are unable to exploit the protocol to get on internal networks.

06 MODEL THE DATA

The corporate-wide deployment and adoption of analytics or IIoT is often delayed by the variability of data coming off the factory floor. From one machine to the next, each industrial device may have its own data model. Historically, vendors, systems integrators, and in-house controls engineers have not focused on creating data standards. They refined the systems and changed the data models over time to suit their needs. This worked for one-off projects, but today's IIoT projects require more scalability.

The first step in modeling data is to define standard models required in the target system to meet the business goals of the project. At the core of the model is the real-time data coming off the machinery and automation equipment. Most of the real-time data points will map to single-source data points. However, when a specific data point does not exist, users can derive data points by executing expressions or logic using other data points. They also can parse or extract data from other data fields, or add sensors to provide required data.

These models also should include attributes for any descriptive data, which are typically not stored in the industrial devices but are very useful when matching data and evaluating data in the target systems. Descriptive data could be the location of the machine, the asset number of the machine, the unit of measure, the operating ranges, or other contextual information. Once the standard models are created, they should be instantiated for each asset, process, and/or product. This is generally a manual task but can be accelerated if the mapping already exists in Excel or other formats, if there is consistency from device to device that can be copied, or if a learning algorithm can be applied.

07 FLOW THE DATA

When the modeling is complete, the data flows should be controlled model-by-model. Organizations typically perform this by identifying the model to be moved, the target system, and the frequency or trigger for the movement. Over time, data flows will also require monitoring and management.

Wrap Up

Industrial environments change over time. Manufacturers replace equipment, change programs, redesign products, upgrade systems, and new users need new information to perform their jobs. Amid this change, OT and IT professionals will collaborate on new projects aimed at improving factory floor productivity, efficiency, and safety. They will need industrial data that's fit for purpose to make use of it. And they will need tools—like a DataOps integration hub—to help them accomplish this task at scale. By using an integration hub, administrators can evaluate equipment and system changes, and identify integrations they must modify or replace. They can make changes to data models and enable new flows in real time.

Making industrial data fit for purpose will be critical to manufacturers looking to scale their IIoT projects and wrangle data governance this year.

A Glossary for Industrial DataOps

AGGREGATION

A consolidated view in the property set of all attribute and variable data from source PLCs, machine controllers, RTUs, smart sensors, and other systems.

ATTRIBUTE

A data characteristic from a source PLC, machine controller, RTU, smart sensor, or other system with a static value (like location) or a dynamic value (like temperature or machine state).

CONNECTION

A connection in an Industrial DataOps solution represents a path to a source system that contains inputs and outputs. An input represents a path to a data point contained in a connection that can be read. An output represents a path to a data point contained in a connection that can be written to.

CONTEXTUALIZATION

Data structures on PLCs and machine controllers have minimal descriptive information—if any. In many cases, data points are referenced with cryptic data point naming schemes or references to memory locations. Contextualization transforms raw data into information by presenting human-readable property names and adding static metadata to the data set. Contextualization enables industrial data to be used more easily outside of the controls environment for machine maintenance, process optimization, quality, and traceability.

CORRELATION

The purpose of automation data has historically been to control and monitor the production process. Therefore, industrial data is correlated for process control. In cases where industrial data must be analyzed and aligned by machine for predictive maintenance, by process for process optimization, or by product for quality and traceability, the data must be assembled and contextualized appropriately for each use case before it can be used. Correlation prepares information for its end purpose by assembling, contextualizing and transposing the data into a usable state.

DATA MODEL

The data model forms the basis for standardizing data across a wide range of raw input data. A data model is comprised of a collection of attributes that are common to the logical item. When working with industrial data, a data model is typically a standard representation of an asset, process, product, system, or role.

From one machine to the next, each industrial device may have its own data model. Historically, vendors, systems integrators, and in-house controls engineers have not focused on creating data standards. They refined the systems and changed the data models over time to suit their needs. This worked for one-off projects, but today's IIoT projects require more scalability.

To handle the scale of hundreds of machines and controllers—and tens of thousands of data points—a set of standard models can be established within an Industrial DataOps solution. The models correlate the data by machinery, process, and product and present it to the consuming applications. This systematic approach of building data models greatly accelerates the usage of this information and simplifies the management of the integrations.

At the core of the model is the real-time data coming off the machinery and automation equipment. This data must often be augmented from many sources, including other equipment or controllers nearby, smart devices or sensors, derivations or transformations computed from existing data points available, metadata manually entered, and data from other databases or systems. Once the standard models are created in the Industrial DataOps solution, they can be instantiated for each logical asset, process, and/or product.

DATA PAYLOAD

Data payload is the collection of data that is assembled through the model instance and is sent out in the flow to the target connection. This payload must be formulated in a way the target system can consume it and must include enough information such that it is understandable, identifiable, and useful.

EDGE

Computing and data handling may be done at the edge, in on-premises servers, or in the cloud.

The edge refers to the end of the TCP/IP network where it connects to industrial automation equipment. The edge also includes the computers installed close to this network boundary that influence the data flowing through it. As computers have become less expensive, easier to manage, and more robust, the prevalence of system architectures with minimally configured computers installed at the edge to process, route, and analyze data has increased. Edge-located computers reduce latency and increase response time, making them ideal self-contained cells to support an Industrial DataOps solution. These computers range from PLCs and network switches with open Linux or Windows cores, to single board computers like Raspberry Pi, to PC-sized industrialized computers.

ETL

ETL (Extract, Transform, Load) solutions integrate business systems with analytics systems. ETL solutions are designed to extract data in a batch process from systems and databases like CRM and ERP, combine this data in an intermediate data store, and then provide tools to manually and automatically transform the data by cleaning it, aligning it, and normalizing it. The data is then loaded into a final data store to be utilized by analytics, trending, and search tools.

Traditional ETL falls short for industrial data for a number of reasons. For one, the data must be processed in real time not in batches. Also, the data models must be standardized since each machine has its own data definitions, which generate a higher volume of data models than is typically processed in ETL solutions for business systems. And, contextualization is critical and more extensive due to the source devices and protocols.

Typical users of ETL solutions for business systems are IT or data science professionals. However, for industrial use cases, the OT team must define and maintain data cleanup since they're familiar with the source systems, data nuances, and changes to the automation equipment.

FLOW

A flow defines the mapping and execution of data coming in and data going out of an Industrial DataOps solution. A flow's source can be a simple primitive input or a modeled instance, while its target is an output. Data flows may be controlled model-by-model by identifying the model to be moved, the target system, and the frequency or trigger for the movement.

INSTANCE

Models are leveraged through the creation of a model instance, each of which are unique to a specific instance of an asset, process, product, system, or role. Whereas a model specifies the standard attributes of a type of asset, process, system, or role, a model instance represents one of these items with mappings to actual live data. For example, a model may be created to represent how a quality manager's view of a manufacturing line will be standardized. If there are 10 manufacturing lines, 10 model instances would be created and populated with data to represent each one.

METADATA

Metadata is data about data. For instance, metadata on a pressure gauge could include the unit of measure for the pressure value or metadata on a machine could include where it is located on the factory floor or the make and model of the machine.

NORMALIZATION

Normalization requires converting property values to common units of measure, like converting a temperature value from Fahrenheit to Celsius or converting a temperature sensor's raw, unitless measurement range to degrees Celsius.

Normalization can also be applied to data flow. For example, analytics systems typically expect to receive data at a consistent or normalized frequency. An Industrial DataOps solution can align data after it has been collected and normalize its flow to consuming applications.

STANDARDIZATION

The automation in a factory evolves over time with machinery and equipment sourced from a wide variety of hardware vendors. This variety in machinery results in a wide range of available data. Some data points may simply have different names, while others may have different units of measure or different measurements entirely. Standardization enables the user to homogenize the property set by asset, process, product, or target system, which allows the data to be rapidly adopted in analytics, visualization, and other systems.

TRANSFORMATION

Industrial computing devices like PLCs, machine controllers, smart sensors, and embedded devices typically represent values and state with shorthand abbreviations or numbers. While this format is ideal for storage and coding, it is not usable by anyone who is not intimately familiar with the programming of the device. From one machine to the next, unique transformation must be defined and performed. For example, 1 = RUNNING.

Transformations may also include statistical calculations of raw data like the average, min, and max temperature values checked every second but recorded every hour. Transformations may also be used to derive an attribute value when a device does not have a unique data tag for it.

UNIFIED NAMESPACE

A consolidated, abstracted structure by which all business applications are able to consume real-time industrial data in a consistent manner. The benefits of a unified namespace (UNS) include reduced time to implement new integrations, reduced efforts to maintain data integrations, improved agility of integrations, access to new data, and improved data quality and security.

Appendix

About HighByte

HighByte is an industrial software development company in Portland, Maine, building solutions that address the data architecture and integration challenges created by Industry 4.0. We believe contextualized and standardized data is essential for Industry 4.0 to reach broad adoption. That's why we've launched HighByte Intelligence Hub—enabling manufacturers to securely connect, model, and flow valuable industrial data throughout their extended enterprise without writing or maintaining code. HighByte Intelligence Hub is the first DataOps solution purpose-built to meet the unique requirements of industrial assets, products, processes, and systems at the Edge.

About HighByte Intelligence Hub

HighByte Intelligence Hub is the first Industrial DataOps solution designed specifically for Operations Technology teams. Purpose-built to model and manage plant floor data at the Edge, the software enables manufacturers to securely connect, merge, model, and flow valuable industrial data throughout the extended enterprise without writing or maintaining code. HighByte Intelligence Hub provides the critical data infrastructure for Industrial Transformation. Learn more and request a free trial at **https://highbyte.com.**

Author Biographies

TONY PAINE

Tony Paine is the Chief Executive Officer of HighByte, focused on the company's vision. He led the architecture and development of the company's initial product offering, HighByte Intelligence Hub, and continues to be instrumental in its design.

Tony's passion for integrating software and hardware began in early childhood when he developed an application that turned a rudimentary text editor into a word processor with generic print capabilities. He focused his education on this interest and earned a Bachelor of Science in Electrical Engineering with a concentration in Computer Software and Hardware Design from the University of Maine in 1996. Tony's entrepreneurial spirit and thirst for knowledge brought him back to the University of Maine to earn a Master of Business Administration with a concentration in Data Analytics in 2021.

For the past 20 years, Tony immersed himself in industrial software development and strategy at Kepware, most recently serving as CEO. He led the company through a successful acquisition to PTC in 2016 prior to founding HighByte in 2018. Tony has contributed to a variety of technical working groups, helping to shape the direction of standards used within the automation industry.

Connect with Tony on **LinkedIn®** professional networking services.

JOHN HARRINGTON

John Harrington is the Chief Business Officer of HighByte, focused on defining the company's business and product strategy. His areas of responsibility include market research, customer use cases, product priorities, go-to-market, financial planning, and sales.

John is passionate about delivering technology that improves productivity and safety in manufacturing and industrial environments. He has spent his 25-year career both delivering software to manufacturers and working for manufacturers in operations roles. This experience has given him a unique perspective on how suppliers and end users each play an integral role in implementing new technology solutions.

John has a Master in Business Administration from Babson College and a Bachelor of Science in Mechanical Engineering from Worcester Polytechnic Institute.

Connect with John on **LinkedIn**® professional networking services.

ARON SEMLE

Aron Semle is the Chief Technology Officer of HighByte, helping to guide technology and product strategy through product development, providing technical evangelism, and supporting customer success during the pre-sales, post-sales, and renewal cycle.

Aron previously worked at Kepware and PTC from 2008 until 2018 in a variety of roles including software engineer, product manager, R&D lead, and director of solutions management, helping to shape the company's strategy in the manufacturing operations market. For the past two years, Aron has worked as an entrepreneur and co-founder of upBed, a Maine-based startup developing technology to provide autonomy and person-centered care for elderly populations. He joined HighByte as CTO in September 2020.

Aron has a bachelor's degree in Computer Engineering from the University of Maine, Orono.

Connect with Aron on **LinkedIn®** professional networking services.

TOREY PENROD-CAMBRA

Torey Penrod-Cambra is the Chief Marketing Officer of HighByte, focused on the company's market presence and ability to operationalize. Her areas of responsibility include brand development, media relations, demand generation, and product marketing. She also oversees funding and investor relations for HighByte.

Torey is a marketing professional with nearly 15 years of experience creating compelling brand experiences that drive customer acquisition and expansion in highly technical environments. Torey's career began with a focus on biotechnology and international pharmaceutical product launches, and then evolved into a fast-climbing career in B2B industrial software. She is passionate about securing equal STEM opportunities for women, and excited by the potential of the Internet of Things in industrial environments.

Torey applies an analytical, data-driven approach to marketing that reflects her academic achievements in both chemistry and ethics. Torey received a Bachelor of Arts in Chemistry from Miami University in Oxford, Ohio and completed post-graduate studies in Medical Ethics at the University of Pittsburgh.

Connect with Torey on **LinkedIn®** professional networking services.

Special thanks to Jonathan Katz, freelance writer and owner of
JSK Communications LLC, for editing this piece.