SAE INTERNATIONAL

SAE/USCAR-53 REVISION 0

Issued 2023-06-01
Revised Initial Release
2023-07-07

INDUSTRIAL DATA COMMUNICATION FOR AUTOMOTIVE MANUFACTURING

Notice about interim revisions: editorial updates may be made as “interim revisions.” Interim revisions are documented as “revision letters” and are available online on the website, https://uscar.org/ewcapbak/spec-revision-letters/.

TABLE OF CONTENTS

QUESTIONS REGARDING THIS DOCUMENT: (248) 273-2470 FAX (248) 273-2494`
TO PLACE A DOCUMENT ORDER: (724) 776-4970 FAX (724) 776-0790

1. SCOPE

The provisions of this document, attachments, and supplements apply to the creation of a universal methodology for data communication, machine language, and reporting to and from manufacturing devices in the automotive production (Smart Manufacturing) environment.

Universal connectivity is one of the foundational technologies enabling data sharing among participating components of an Industrial Internet of Things (IIoT) system. Connectivity provides the ability to exchange data among participants within a functional domain, across functional domains within a system, and across systems. The data exchanged may include sensor updates, events, alarms, status changes, commands, and configuration updates. Connectivity is a crosscutting function in the Industrial Internet Reference Architecture. It provides the ability to exchange data between participants within and across functional domains (control, operations, information, applications, and business).

2. INTRODUCTION AND BACKGROUND

For automotive companies to remain competitive in the era of Smart Manufacturing, data connectivity and integration between systems must utilize open standards. Supporting legacy proprietary closed systems on the manufacturing floor has created an unacceptable burden of lost time, talents, and profit.

The overarching goal of IIoT connectivity is to unlock data in isolated systems (“silos”) to enable data sharing and interoperability between previously closed components and subsystems (brownfield), and new applications (greenfield), within and across industries.

The makers of industrial control equipment currently in use within the automotive industry fall along a continuum of readiness to adopt open standards. To determine the feasibility of establishing a machine data communication standard, a benchmarking exercise was conducted.

Twenty manufacturing equipment OEMs, eleven entities that comprise standards organizations and companies from other relevant industries, plus several industry-recognized subject matter experts were surveyed and benchmarked as inputs to the Concept Feasibility Milestone of a machine data communications standard.

Some of the largest makers of automation controllers actively resist the adoption of open standards as it threatens their longstanding revenue model. Conversely, a market group of process controller companies is aggressively pursuing open standards and innovative product strategies to meet the requirements of Smart Manufacturing. Throughout this landscape two primary communication protocols dominate the marketplace: MQTT (**M**essage **Q**ueuing **T**elemetry **T**ransport: featured by innovative process controllers) and OPC UA (**O**pen **P**latform **C**ommunication **U**nified **A**rchitecture: favored by legacy automation controllers).

A number of integrators and systems experts, acknowledge both these protocols will exist for the foreseeable future, so therefore, efforts to standardize should be focused at the data syntax level, not on the protocol. The widely popular Sparkplug B data standard was proposed for consideration, and after internal discussions with their respective companies the team determined that it needed to develop its own standard starting in 2021.

The *Figure 1* diagram depicts an overarching future state for Industrial Data that shows data flowing to and from the Industrial Data Sources through the MQTT Broker to and from the Smart Factory Solutions to be implemented.

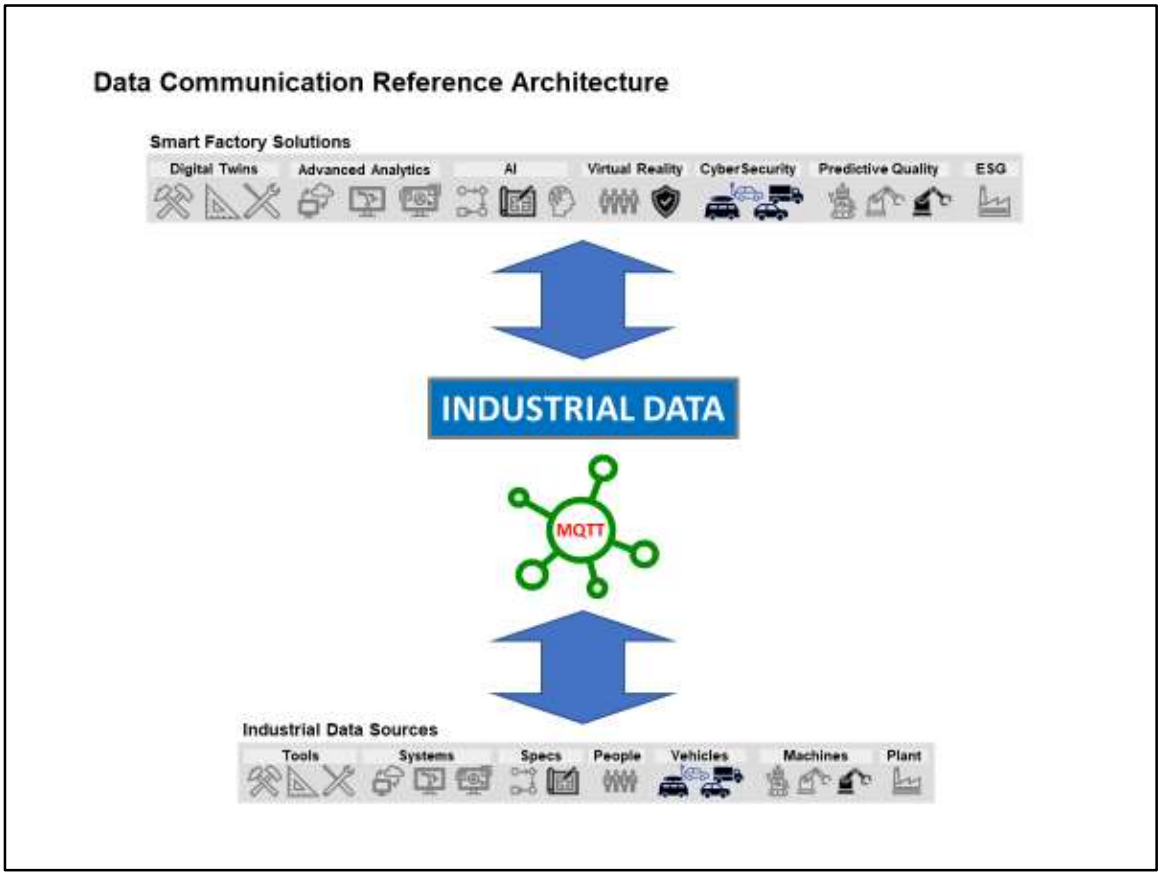

Figure 1 – Data Communication Reference Architecture

2.1 Smart Manufacturing

Smart Manufacturing requires a high degree of automation and integration of systems on the shop floor to deliver optimal performance and ROI.

- These systems have evolved over decades in a highly proprietary, closed manner by the companies who design, build, and sell them.
- The resulting landscape of incompatible machine communication creates costly delays and complexity in product launches for automotive companies.
- Interoperability and extensibility of these machines is imperative to insure cost efficient operation and competitive viability.

2.2 Manufacturing Devices

Multiple devices present data to IT applications in varied and sometimes proprietary methods as defined by the equipment vendors.

- Collection of data from these devices is typically developed from scratch for each new device type or vendor.
- Some of the methods are inherently insecure due to the age or processing capabilities of the devices.
- Some methods are intentionally closed to enable vendors to sell additional software products or analytics services (typically cloud-based SaaS).

3. OBJECTIVES FOR THIS STANDARD

This is a standard specification for Operational Technology (OT) equipment data publishing to be included with Equipment Purchasing specifications Statement of Requirements (SOR). This is a specific Global Standard for the automotive industry and provides the ability to:

- Allow all new equipment to easily plug into manufacturing data analytic tools.
- Specify the Event Types for when data should be created and sent.
- Outline requirements for communication methods implemented and describe the data needed to perform analytics.
- Specify data formats within schemas.
- Devices that shall provide the capability to be configured with the MQTT brokers topics to which they publish and subscribe. Refer to [4].
- Specify Data transport and protocols for consumption by manufacturing data analytic tools.

This document defines the Common Data Model (CDM) Standard Specifications for structuring data from Industrial Control Systems such as Process and Automation Controllers, Sensors, and Actuators in a standard format so that a common IT and Controls solution aligns for all types of data to and from production manufacturing equipment and devices.

This specification provides general data communication requirements. This document shall be followed for all applications and protocols developed for the automotive industry Global standard compliance. This is an overarching standard that defines basic data communication methods but also allows OEMs and suppliers to have additional specifications.

4. INDUSTRIAL MACHINE DATA COMMUNICATION STANDARDS

Industrial Control System Devices including Automation Controllers (i.e., PLC, NC, Robot Controllers), Process Controllers (i.e., Tooling Controllers, Vision, Test), Sensors (i.e., vibration, temperature, current, pressure), and Sensor systems for Maintenance shall support:

4.1 Native Support for Open IIoT Protocols

- Pub/Sub; MQTT (Preferred) Client/Server
- OPC UA (by Exception)

4.2 Native Support for Encoding Dynamic Message Payload Formats

- JSON
- XML (by Exception)

4.3 Configuration Support for Customer Namespace

- Tools to Import Customer specified Namespaces
- Tools for Mapping Device Namespace to Customer Namespace

4.4 MQTT Topics

Devices shall provide the capability to be configured with the MQTT brokers topics to which they publish and subscribe. Refer to [4].

Emerging Best Practice Architecture

Industrial information is sent via the MQTT architecture as described in *Figure 2*. This enables multiple Enterprise systems to subscribe and ingest various manufacturing data via a pub/sub system known as MQTT. This allows devices to publish data to a broker and applications to subscribe to that data.

The focus of this Standard is the portion of the diagram in *Figure 2* to the right of the MQTT broker. Any plant floor device whether legacy or Industrial IoT compliant communicates to an MQTT broker using JSON formatted payloads using the common data structure detailed in this Standard.

All Asset types shall publish its data in a JSON encoded payload to an MQTT broker, directly or via an edge gateway for legacy applications. The communication protocol between plant floor devices sending machine data to the collection system utilizes an MQTT transport via TCP/IP. All PLC and plant floor devices that publish messages shall be time synced to the central timeserver available to the plant. In addition to the Plant floor device sending time stamp data, a time stamp of the data insertion into the database is also stored.

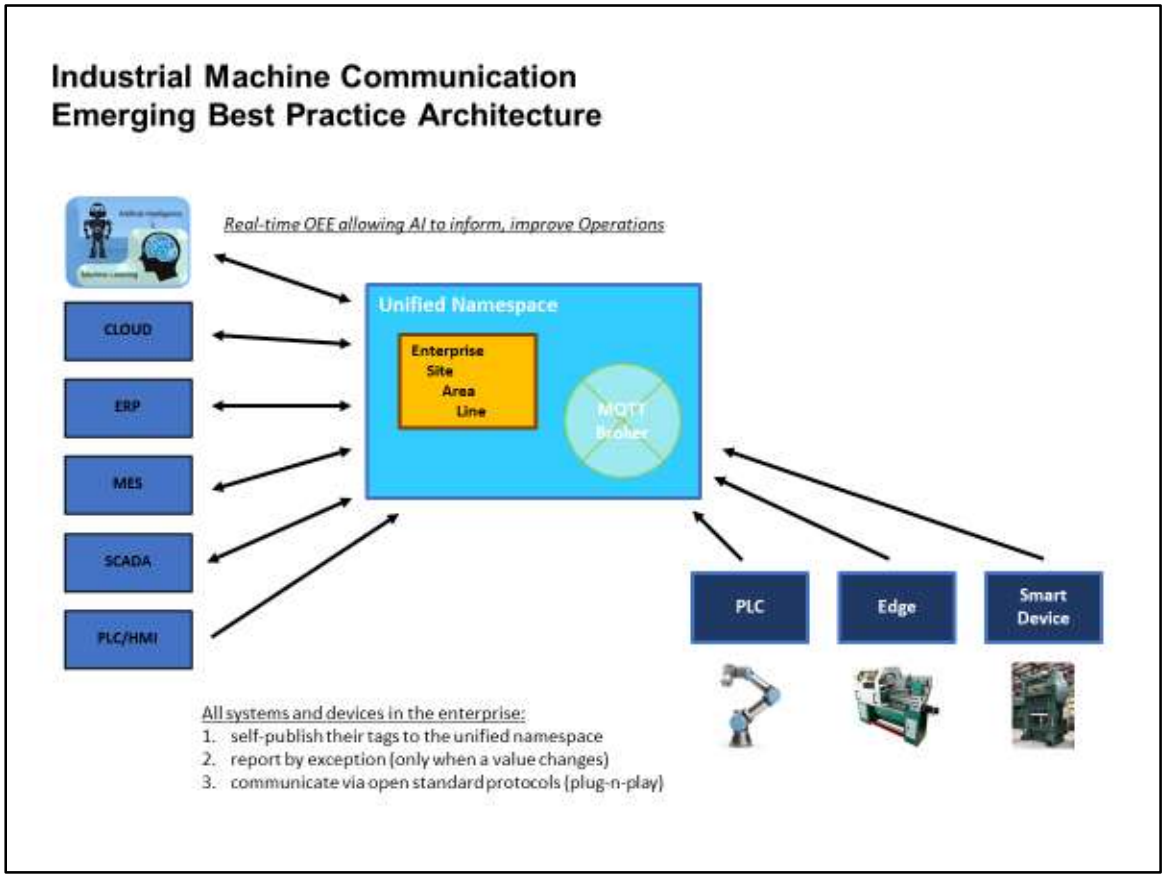

Figure 2 – Emerging Best Practice Architecture

4.5 The Imperative for an Overarching Industrial Data Communication Standard

Smart Manufacturing requires a high degree of automation and integration of systems on the shop floor to deliver optimal performance and ROI. These systems have evolved over decades in a highly proprietary, closed manner by the companies who design and sell them. The resulting landscape of incompatible machine communication creates costly delays and complexity in product launches for automotive companies. Interoperability and extensibility of these machines is imperative to insure cost efficient operation and competitive viability.

After benchmarking the industry and evaluating existing IIoT standards initiatives, it became clear that a standard was needed that met the requirements of the Automotive Industry OEMs and Suppliers.

Figure 3 depicts the various levels of the industrial machine communication landscape; and while data communication standards are being developed within these various levels, the standards typically focus on the horizontal communication within the level. The challenge for the Automotive Manufacturers is the need for seamless data communication across the levels, as well as vertically throughout the industrial landscape.

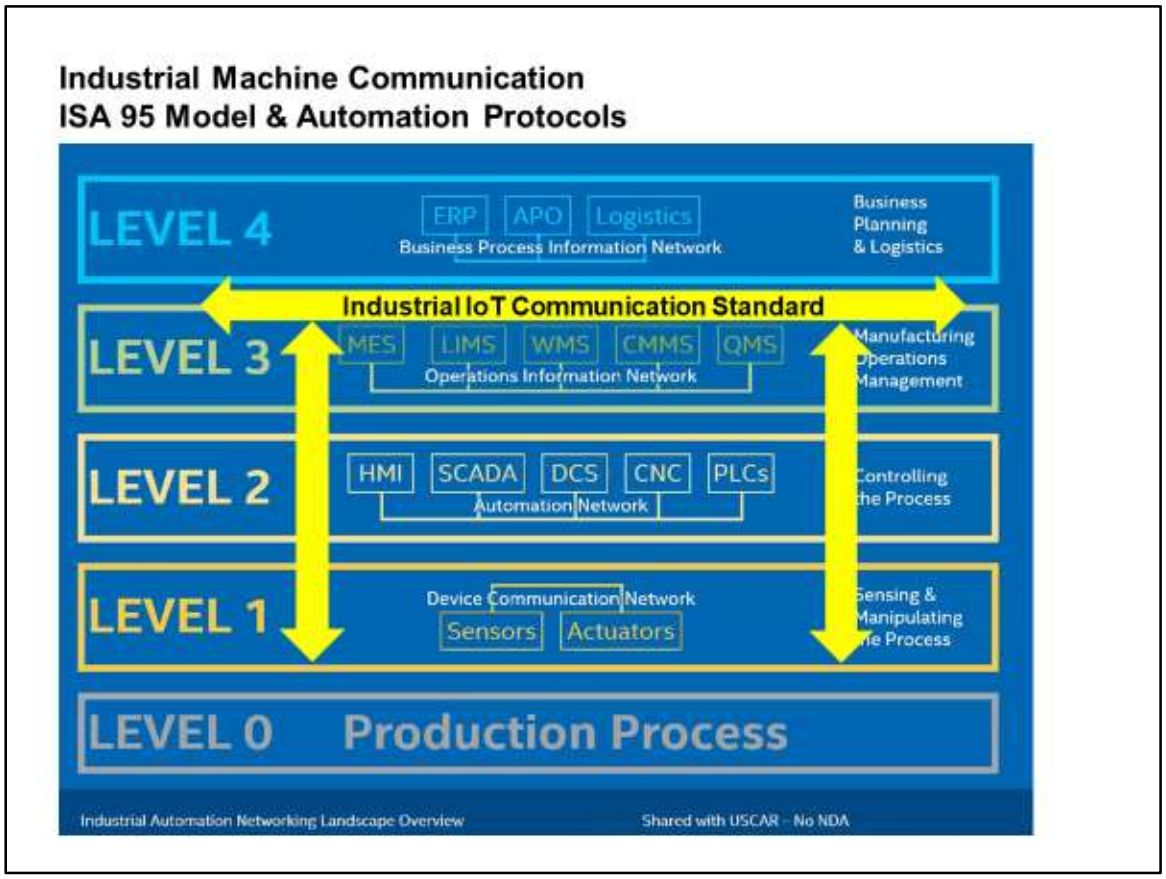

Figure 3 – Industrial IoT Communication Standard

5. INDUSTRIAL DATA ARCHITECTURE

Data Architecture and communication within a Smart Manufacturing environment allows for many different types of data along with different ways in which data is communicated from and to the manufacturing floor. A broad data classification diagram as shown in Figure 4 denotes the variety of data that may be communicated within this environment.

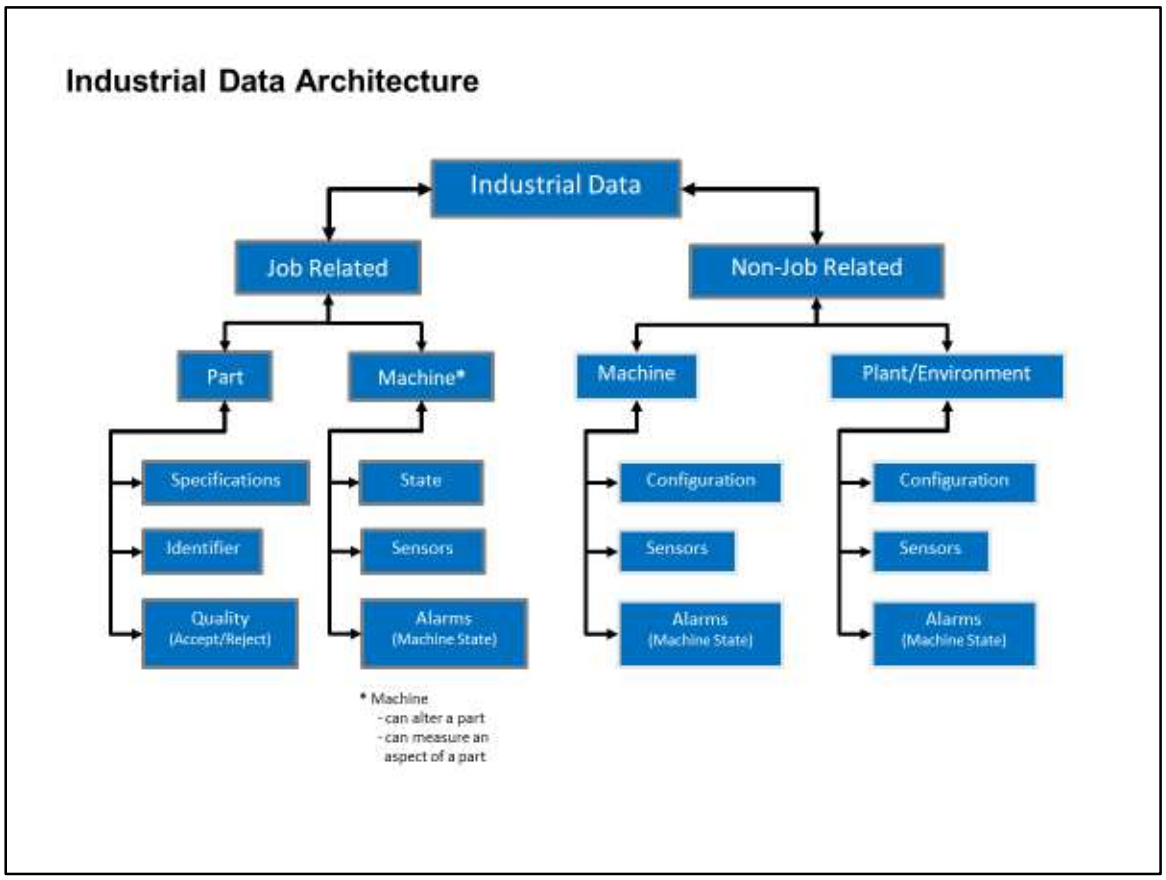

Figure 4 – Industrial Data Architecture

5.1 Industrial Data Classification

There are two kinds of Industrial Data: Job Related and Non-Job Related.

5.1.1 Job-Related

“Job” is defined as the act of making a part in production. Production of a part involves the part itself and the machine that acts on the part. A “machine” is defined as the entity that is capable of either modifying a part or measuring an attribute of it. Also, job related data can be categorized into Part and Machine data.

A part data further contains identifier, specification, and quality information. The part identifier is a unique entity that identifies a part, (e.g., a serial number). The specification data pertains to the requirements to which a part is to be made, (e.g., dimensions, tolerances, and any other metadata related to the part). After an action is taken on a part, it is expected to pass certain quality checks. These values could be quality attributes or a Boolean indicating pass or fail.

A machine has data pertaining to both job related and non-job related. The type of data reported is different in each category. The job related machine data is available when it is actively altering or making a part, and can include machine state, data from the sensor attached to the machine or an alarm. The state communicates the operating mode it is in with “idle” as a valid state and should technically indicate the associated data is not job related. When a machine is not idle, it is expected

to record data from its sensors. These data may be (rarely) scalars, time-series signals, images, or video. When data is scalar, it could fetch a place in a database but the other types of data require some file format to package the data.

Unlike sensor data, an alarm is usually thought of as a Boolean. However, additional information is necessary to describe the alarm. An alarm is triggered if the data from a sensor exceeds some threshold set by the configuration of the machine. So, the sensor on which the alarm was triggered and the time-stamp at which such an event took place must be packaged with this data.

5.1.2 Non-Job Related

The non-job related machine data is obtained when the machine is in “idle” state. This is the opportunity to communicate the configuration(s) of the machine. It could contain a library of operating configurations and the conditions that require any of those. Just because a machine is idle does not mean it does not have information to provide. The sensors in a machine could be active and report data periodically to keep track of any condition of interest. Such information can help in tearing readings in a non-idle state. Machine calibration could also be one of those non-job related sensor data opportunities. Machine alarms in this state would generally be indicative of maintenance and tool/sensor change.

The Plant/environment data is essential to keep track of the overall conditions for manufacturing. For example in a dryroom, it is essential to monitor humidity and temperature at all times.

The environmental requirements need to be known for the given manufacturing process. Such information is captured in the configuration data for the plant. The need for sensors then becomes a necessity and data from it would need to be communicated in a set schedule.

Any violations of the desired configuration, as indicated by any sensor, would need to be recorded as an alarm with all the necessary additional information, such as the sensor that triggered and the timestamp of the event as a minimum.

6. EVENTS THAT TRIGGER COMMUNICATIONS

6.1 Process Events

6.1.1 Cycle Data (Refer to Machine Data Schema and Part Data Schema)

Cycle data consists of one or more messages describing the measurements and activities of the most recent job (e.g., weld or torque).

6.1.2 Continuous Data (Refer to Sensor Data Schema)

Devices collect measurements at a configurable frequency and generate messages containing a configurable number of measurements. Multiple measures can be included in a single message (e.g., vibration, motor current, or temperature).

6.2 Other Normal Events (Refer to Machine Data Schema)

An event message may be generated to signal a normal non-job-related event and any measures recorded during the event (e.g., Barrel Change, Tip Dress).

6.3 Device Events (Refer to Machine State Schema)

6.3.1 Alarms/Alerts

Alarm or alert messages may be generated to signal an unexpected condition (faults, warnings, info).

6.3.2 Device State (Refer to Machine State Schema and Heartbeat)

A configurable periodic message may be generated related to the current state of the device.

6.3.3 Configuration Changes (Refer to Machine Configuration Schema)

One or more messages may be generated describing edits to the configuration. These should include detailed information relating to the specific item that changed, previous and new values, and the user who made the change.

7. MESSAGE SCHEMAS

Each message type has its own independently versioned schema.

7.1 Philosophy

Schemas follow a major.minor-patch version philosophy.

7.1.1 Major version

A Major version is incremented with a change that breaks backward compatibility such as a change in data format, tighter value restrictions, deleting keys, or values from an enumeration list or required parameter setting.

7.1.2 Minor version

A Minor version is incremented with a change that maintains backward compatibility such as the addition of key:value pairs, loosening value restrictions, or adding values to enumeration lists.

7.1.3 Patch version

A Patch version is incremented with changes that are bug fixes, descriptions updates, or other changes that are done in a controlled manner so as to not risk message validation issues.

7.2 Increments

It is possible that schema versions are incremented by more than one on occasions where it is desirable to realign the versions across multiple messages. For similar reasons, schema initial releases may have a version greater than 001.

7.3 Schema Properties and Definitions

Schemas detail the required format of the JSON message including the nesting of data, valid data keys, and their respective properties. Properties include but are not limited to data types, required fields, minimum and maximum values, minimum and maximum string lengths, as well as valid characters and formats. All properties listed in the respective schema are followed. For information on JSON properties, refer to [1].

7.3.1 Description

The property name is: “description” which is a brief description of the object.

7.3.2 Data Type

The property name is: “type”. Type specifies what data types may be reported against this key.

7.3.3 Required Fields

The property name is: “required”. If a key is listed in the schema as required, then it shall be reported whenever the parent object is reported.

7.3.4 Minimum Numerical Value

This property name is: “minimum”. The numerical value reported shall be greater than or equal to this limit (the minimum value).

7.3.5 Maximum Numerical Value

This property name is: “maximum”. The numerical value reported shall be less than or equal to this limit (the maximum value).

7.3.6 Minimum String Length

This property name is: “minLength”. The length of the string reported must be greater than or equal to this limit.

7.3.7 Maximum String Length

This property name is: “maxLength”. The length of the string reported must be less than or equal to this limit.

7.3.8 Enumerated Values

This property name is: “enum”. The value reported must match a value in the enum list of the schema.

7.3.9 String Format

This property name is: “pattern”. The string reported must match the pattern listed in the schema. For information on JSON schema patterns, refer to [2].

7.3.10 Examples

This property name is: “examples”. This property is only for the purpose of showing examples and has no requirements associated with it.

7.3.11 Default

This property name is: “default”. This property is only for the purpose of defining a default value and has no requirements associated with it.

7.4 Manufacturing Message Structure

7.4.1 Payload

The message payload consists of a message header and the message body. The message header gives basic information such as: message timestamp, source of message, and message payload type.

7.4.2 Flat

Messages are flat and do not contain tabs, carriage returns, or any other unnecessary white space.

7.4.3 Keys

Keys not listed in the schema are not used unless the schema specifically calls out “additionalProperties”:true for the object in which the key is included.

A key listed in the schema as required is used whenever its parent object is used.

Keys are comprised of only upper- and lower-case characters, digits 0-9, and contain no spaces or special characters.

Keys contain an upper-case letter for the first character. Numbers are not permitted for the first character.

Keys are in PascalCase (where the first character of each word is upper-case with the remaining characters being lower-case and containing no spaces or separators).

Values are reported according to the respective schema.

Non-required keys with empty or null values may be omitted from the message as they occupy storage space without adding value.

Non-required keys with unchanged data may be omitted from the message.

7.4.4 Message Payload

Every message type's payload includes a Message Header and the Message Body.

7.4.4.1 Message Header

The header consists of the message timestamp, schema version, message type, message sub-type, and IIoT asset DeviceID as defined in the respective schema.

7.4.4.1.1 MessageTimeStamp

MessageTimeStamp is populated with the Universal Time Coordinated/Universal Coordinated Time (UTC) time when the message is initiated using the extended format with separators: hyphen between year, month, and day; the letter T between date and time; with a colon between hour, minute, and second; period to separate seconds from milliseconds, ending with the letter Z (i.e., "2020-12-31T23:59:59.999Z"). Milliseconds are recommended.

7.4.4.1.2 SchemaVersion

SchemaVersion is populated with the schema major.minor version used for the message. This is used for message validation and during the ingestion of the message data into the database.

7.4.4.1.3 MessageType

MessageType is populated with one of the predetermined types each having their own schema.

7.4.4.1.4 SubType

SubType is populated with one of the predefined sub-types for that schema. The sub-type identifies the focus or purpose of the message. For example, a MachineData message with a sub-type StaStart identifies that it is the start of a new cycle and the part type/ID specifics known at that time.

7.4.4.1.5 DeviceID

Publisher is the device ID name of the smart device and/or application that is publishing the message.

The DeviceID is comprised of only upper- and lower-case characters, digits 0-9, hyphens and underscores, and contains no spaces or other special characters. The device should allow for a user created ID to be configured.

DeviceID shall be unique within the world.

7.4.4.2 Message Body

The body consists of transaction counter, cycle identifier when listed as required in the schema, followed by manufacturing data. The message body content does not require all data listed in the schema to be populated unless it is specifically stated in this specification or in the schema itself.

7.4.4.2.1 Transaction Counter

TransCounter is an independent counter for each message type that is incremented by one for each message send. The counter is designed to rollover at a reasonable value and is contained in a 32-bit signed integer.

7.4.4.2.2 Cycle Identifier

CycleID is a common identifier used to link different messages that happen during the same cycle and/or before the next cycle.

The CycleID is populated with the UTC timestamp at the start of every part cycle (i.e., part load) and is never blank.

The CycleID is populated regardless of station cycle state (manual or automatic cycling).

The CycleID is not reset by changes in a station's mode (auto, manual, cycle on, or cycle off).

Devices associated to a machine's cycle that do not manage the CycleID receive the CycleID from the main controller.

Devices not able to receive the CycleID from the main controller that manages the CycleID do not generate their own CycleID. Instead the device reports a null CycleID and it is associated to the correct CycleID at a higher level within the system at a later time.

The CycleID uses the extended time format with separators: hyphen between year, month, and day; the letter T between date and time; with colon between hour, minute, and second; period to separate seconds from milliseconds, followed by the letter Z (i.e., "2020-12-31T23:59:59.999Z").

The CycleID is also used by the application to establish the timestamp of events and tasks by adding the event or task start time reported (offset from start of cycle) to the CycleID.

7.4.5 Labels

All values reported with a key "Label" are unique within the object that it is being reported. Additional requirements on uniqueness are included in the schema description for the specific Label key. These requirements exist because the Label value is used in the storage of associated data reported as arrays.

All values reported with a key "Label" are comprised of only upper- and lower-case characters, digits 0-9, underscores, and contain no spaces or other special characters.

The length of a Label does not exceed 16 characters unless specified otherwise in the schema.

7.4.6 Relative Time

Start, End, and Duration of an event or task can be reported along with the respective design time, a warning limit, and a fault limit.

7.4.6.1 Start and End Times

Start and End times reported are an offset in seconds with millisecond resolution to three (3) decimal places (e.g., 59.999) from the start of cycle timestamp reported as the CycleID in that message.

Ingestion and analysis applications can later add the offset to the CycleID to establish the timestamp of the event or task, producing the benefit of easily overlaying cycle events and tasks regardless of the actual time they started.

Events that are not part of a regular cycle could occur between or span multiple production cycles. For example, purge, blow-off, vacuum pump, machine lube, and tip dress events can happen during the cycle but can also happen between cycles or span cycles. When reporting such events, the Start and End times reported are an offset from the start of cycle reported as the CycleID in the message.

Because it is possible that an event could start during one cycle and finish during a different cycle, it is critical that the Start and End times reported are offsets from the CycleID of the message in which they are reported. If the CycleID of the message is the start time of a cycle that started after the event started, the Start time reported is a negative number such that when it is added to the CycleID results in the correct time for the start of the event.

7.4.6.2 Duration

Duration is the time between the start and end-time and may be expressed in hour, minute, second, and millisecond.

7.4.6.3 Warning Limits

The warning limits are upper and lower preset time values, that when the duration hits the limit, a warning is generated.

7.4.6.4 Fault Limits

The fault limits are upper and lower preset time values, that when the duration hits the limit, a fault is generated.

7.4.7 Timestamps:

Timestamps reported in the payload are populated with Universal Time Coordinated/Universal Coordinated Time (UTC) time.

Timestamps use the extended time format with separators: hyphen between year, month, and day; the letter T between date and time; with colon between hour, minute, and second; period to separate seconds from milliseconds, followed by the letter Z (i.e., “2020-12-31T23:59:59.999Z”). Milliseconds are recommended.

7.4.8 Features and Feature Arrays:

The Feature object is used to report a single measurement, calculated value, or text, where a FeatureArray object is used to report that measurement, calculated value, or text collected at a constant frequency over a period of time.

A FeatureArray object includes the StartTimeStamp and PollRate to allow the data to be plotted.

Features and FeatureArrays include the Type and Statistic keys, to allow classification of the measurement for analytics and visualizations.

Type reports the classification of the measurement from the enumeration list in the respective schema (e.g., Temperature, Vibration, Torque, Units of Measure).

Statistic reports calculation type used when reporting processed data and reports Raw when it is unprocessed data.

7.5 Message Validation

The programmer is responsible to validate sample messages generated against the respective schema using a JSON validation application.

7.6 Manufacturing Schema Contents

This specification does not identify what data is to be collected from a plant floor device or what can be done with the data that is collected. This specification defines the data packaging and communication method of data transfer.

7.6.1 Device Registry Schema

Where applicable, the Device Registry message is used in the registration of valid publishers and subscribers of messages as well as reporting the devices for which they publish and/or subscribe to data.

Registration may be done by an application instead of the device if it is not capable of performing its own registration.

The Device Registry message shall use Quality of Service (QoS) level 1.

The Device Registry message shall report one of the sub-types:

7.6.1.1 Request

The Request sub-type shall be used by a device or application to establish a data source or to update message types or devices to which an existing data source publishes or subscribes.

A Request message shall be published when a new data source is connected for the first time unless it has been previously registered and no changes in its registration are required.

A Request message shall be published when there are changes to the list of messages to which it publishes or subscribes.

A Request message shall be published when there are changes to the list of devices to which messages are published or subscribed.

7.6.1.2 Response

The Response sub-type shall be used by the registration service in response to a Request message from a device or application.

7.6.2 Machine Configuration Schema

Where applicable, the Machine Configuration message is used to report machine parameters that can be adjusted by the operations/operator from the machine or down to the machine from a supervisory system. Examples of machine configuration data include but are not limited to: Stroke and pressure settings, speed override settings, or adjustable VFD settings.

Typically, the machine configuration message is sent on a monitored parameter setting change but can also be sent for parameter backup.

The Machine Configuration message shall use QoS level 1.

The Machine Configuration message shall report one of the sub-types:

7.6.2.1 Actual

The Actual sub-type shall be used to report the machine configuration parameters to the IIoT system.

An Actual message should only report parameters that have changed but may be used to report all parameters at a reasonable frequency as a backup.

7.6.2.2 Update

The Update sub-type shall be used to send machine configuration parameters to the plant floor devices.

An Update message shall only be sent to update the parameters of the plant floor devices.

7.6.3 Device Metadata Schema

Where applicable, the Device Metadata message is used for reporting details about equipment or devices on the plant floor (e.g., model numbers, serial numbers, firmware versions).

The Device Metadata message shall use QoS level 1.

The Device Metadata message does not have any associated sub-types so it shall report an empty string for the sub-type.

The Device Metadata message shall be published on a data change.

7.6.4 Machine State Schema

Where applicable, the Machine State message is used to report machine states, alerts, and notifications. Examples of machine state data include but are not limited to: blocked, starved, cycling conditions as well as faults, warnings, and other notifications.

The Machine State message shall use QoS level 1.

The Machine State message shall report one of the sub-types:

7.6.4.1 State

The State sub-type shall be used to report current machine state (e.g., blocked, starved).

A state message shall be published on change of state and/or after a defined interval has elapsed since the last state message was sent. Varying frequencies in this range among the publishers decreases simultaneous message triggering.

7.6.4.2 Alert

The Alert sub-type shall be used to report alerts such as faults and warnings.

An alert message shall be published on the rising edge of an alert becoming active with Alert.State reported as Active and again when the alert becomes inactive with Alert.State reported as Reset.

7.6.4.2.1 Notification

The Notification sub-type shall be used to report informative text to be displayed on marquees, paging, and other places.

A Notification message can be published once, whenever a condition arises that requires notification or at reasonable frequency when a condition exists as a reminder.

7.6.5 Machine Data Schema

Where applicable, the Machine Data message is used to report data related to machine health and performance, not related to a specific part/assembly. Examples of machine data include but are not limited to: cycle times, counts, energy usage, event times and details, tool details, or measurements.

The Machine Data message shall use QoS level 1.

The Machine Data message shall report sub-types based on events. Sub-type examples are shown:

7.6.5.1 Transfer In (XferIn)

This message shall be used to establish a new cycle and report part and/or carrier information as well as the CycleID.

An XferIn message shall be published as close as possible to the start of a new part cycle as soon as the desired part and carrier information is known. Typically, the start of the part cycle is when the part/carrier is transferring into the station.

7.6.5.2 Station/Cycle Start (StaStart)

This message shall be used to establish the part has arrived in station and where applicable can be used for multiple purposes such as: to start timing countdowns, update part tracking displays, and trigger inspections.

A StaStart message shall be published as close as possible to the part arriving in station as soon as the desired part and carrier information is known. When the station is not a stop station the message shall be sent when the part has entered the work pitch to the point that work can begin.

7.6.5.3 Events

This message shall be used to report data related to an event such as start/end/duration times, device program numbers, or tool numbers. An event is typically a motion or set of motions.

An Events message should be published on event completion, but event data can be accumulated and published later in the cycle.

Events messages shall be sent within the cycle they are associated with and not accumulated across multiple cycles.

7.6.5.4 Station/Cycle End (StaEnd)

This message shall be used to establish the completion of the cycle and report cycle times and counts as well as other information associated to the overall cycle.

A StaEnd message shall be published at the end of the cycle. Typically, the end of the cycle is when the part/carrier has all conditions for it to be able to release from the station except for downstream trafficking issues.

7.6.5.5 Verification Check (VerCheck)

This message shall be used to report data related to a verification check performed on the machine such as brake checks, homing sync switch checks, and sensor calibration.

A VerCheck message can be published on verification step completion or can be accumulated and published at the completion of the verification process.

7.6.5.6 Waveforms

The Waveforms sub-type shall be used to report waveform/trend data associated to the machine’s health and features associated to those waveforms.

A Waveforms message can contain the waveform data or a retrieval path with enough information for a service to retrieve the waveform data from the device.

A Waveforms message shall be published prior to the next part cycle.

7.6.5.7 Images

The Images sub-type shall be used to report camera images associated with the machine's health and features associated with those images.

An Images message can contain the image or a retrieval path with enough information for a service to retrieve the image data from the device.

An Images message shall be published prior to the next part cycle.

7.6.6 Sensor Data Schema

Where applicable, the Sensor Data message is used to report values or arrays of sensor measurement data. Examples of Sensor Data include but are not limited to: conveyor motor current, pump temperature, and gearbox vibration. The data reported is often independent of a machine or part cycle. When sensor data is too big to communicate (e.g., high resolution vibration) as a part of the payload, the data should be stored in a file and a link to that file is sent as part of the payload.

Typically, the Sensor Data message is sent at a set frequency or after a preset number of readings are recorded, not based on a machine or part cycle.

The Sensor Data message shall use QoS level 1.

The Sensor Data message shall report one of the sub-types:

7.6.6.1 Burst

The Burst sub-type shall be used to report arrays of sensor measurement data.

The measurements shall be taken at a consistent frequency with the poll rate and start time reported along with the data.

A Burst message can be published at a set frequency or after a preset number of readings are collected.

7.6.6.2 Indicator

The Indicator sub-type shall be used to report single values of sensor measurement data. These values are often calculations made from collected burst data.

An Indicator message can be published at a set frequency or after a preset number of readings are collected.

7.6.7 Part Data Schema

Where applicable, the Part Data message is used to report part specific quality data. Examples of Part Data include but are not limited to: task statuses, component unit IDs, part measurements, images and vision system results, rundown and press results, and test system results.

The Part Data message shall use QoS level 1.

The Part Data message shall report one of the sub-types:

7.6.7.1 Part

The Part sub-type shall be used to report details about the part or assembly that is in station (e.g., identifiers, part type/model, status, carrier).

A Part message shall be published as close as possible to the part entering or arriving in station as soon as the desired part and carrier information is known.

7.6.7.2 Traceability

The Traceability sub-type shall be used to report details about sub-components, sub-assemblies, or raw materials that are used in the processing of the current part or assembly. This information can be used in future campaigns if quality issues are found.

A Traceability message shall be published as soon as soon as the information is known.

7.6.7.3 Tasks

The Tasks sub-type shall be used to report data related to process tasks (e.g., status, failures, concerns, start/end/duration times, device program numbers, tool numbers). A task is typically a process step or operation that is performed on the part or assembly.

Tasks messages (data) should be published on task completion, but can be accumulated and published later in the cycle.

Tasks messages shall be sent within the cycle they are associated with and not accumulated across multiple cycles.

7.6.7.4 Concerns

The Concerns sub type shall be used to report quality concerns that need to be tracked.

A Concerns message shall be published prior to releasing the part from the station.

7.6.7.5 Waveforms

The Waveforms sub-type shall be used to report waveform/trend data associated to the part quality and features associated to those waveforms.

A Waveforms message can contain the waveform data or a retrieval path with enough information for a service to retrieve the waveform data from the device.

A Waveforms message shall be published prior to the next part cycle.

7.6.7.6 Images

The Images sub-type shall be used to report camera images associated with the part quality and features associated with those images.

An Images message can contain the image or a retrieval path with enough information for a service to retrieve the image data from the device.

An Images message shall be published prior to the next part cycle.

7.6.8 Heartbeat

Where applicable, the Heartbeat message is used to report a transaction counter sent either at a set frequency or after a set period of message inactivity to inform the system that the device is still communicating.

Typically, the Heartbeat message is sent at a set frequency or after a period of message inactivity.

The Heartbeat message shall use QoS level 1.

The Heartbeat message does not have any associated sub-types so it shall report an empty string for the sub-type.

The Heartbeat message shall be sent after a period of 40 to 50 seconds has elapsed since the last Machine State message with sub-type State or Heartbeat message send. Varying frequencies in this range among the publishers decreases simultaneous message triggering. If State messages are being sent within this frequency, Heartbeat messages are not required.

7.6.9 Maintenance Schema

Where applicable, the Maintenance message is used for maintenance related information such as identifying, requesting, and tracking maintenance actions or identifying maintenance opportunity windows.

The Maintenance message shall use QoS level 1.

The Maintenance message shall report one of the sub types:

7.6.9.1 Ticket

The Ticket sub-type shall be used to request a maintenance ticket be issued to perform a specific task. A maintenance system subscribes to Ticket messages to generate a maintenance ticket.

A Ticket message shall be published when a required maintenance task is identified.

7.6.9.2 Maintenance Opportunity Window (MOW)

The MOW sub-type shall be used to report a calculated time window that a portion of the manufacturing line can be taken out of production without starving the downstream process. This is referred to as a **M**aintenance **O**pportunity **W**indow.

A MOW message can be published on a data change or at a set frequency.

7.6.9.3 Machine

The Machine sub-type shall be used to report that a maintenance task is required on a piece of equipment. Controllers can subscribe to the Machine messages for the purpose of display or switching into a maintenance mode.

A Machine message shall be published when a required maintenance task is identified.

7.6.10 Build Data Schema

Where applicable, the Build Data message is used for reporting build sequence, scheduling, and assembly specific requirements.

The Build Data message shall use QoS level 1.

The Build Data message shall report one of the sub-types:

7.6.10.1 Request

The Request sub-type shall be used by a device or application to request build sequence, scheduling, and assembly specific requirements.

A Request message shall be published when new or updated information is required by the device or application.

7.6.10.2 Response

The Response sub-type shall be used to respond to a Request message from a device or application with the requested information.

A Request message shall be published when a new request is received from a device or application.

7.6.11 Gauge Data Schema

Where applicable, the Gauge Data message is used for reporting part gauging result data from inline gauges, offline gauge benches, CMMs, and other gauges.

The Gauge Data message shall use QoS level 1.

The Gauge Data message shall report one of the sub-types:

7.6.11.1 Internal

The Internal sub-type shall be used to report part quality data from devices (gauges) within the plant.

7.6.11.2 External

The External sub type shall be used to report part quality data from outside suppliers formatted using the GaugeData schema.

7.6.12 Operator Badge Data Schema

The Badge Data Schema may refer to any method of operator identification. Badge Data Schema is used for reporting operator badge-in/badge-out related information if required (optional).

The Badge Data Schema does not have any associated sub-types so it shall report an empty string for the sub-type.

The Badge Data message shall be published on each badge-in or badge-out event.

The Badge Data message shall use QoS level 1.

APPENDIX A - BIBLIOGRAPHY

[1] https://json-schema.org/understanding-json-schema/reference/object.html

[2] https://json-schema.org/understanding-json-schema/reference/regular_expressions.html

[3] https://uscar.org/ewcapbak/spec-revision-letters/

[4] https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf

APPENDIX B - TERMS

EXTERNAL CONSTRAINT – MUST (MUST NOT): The words “must” or “must not” denote a constraint. “Must” or “Must not” is used whenever there is a constraint or obligation, typically due to one or more legal requirements or laws of nature and is to be fulfilled from which no deviation is permitted. Its use shall not be avoided on the grounds that compliance with the standard is considered voluntary.

REQUIREMENT – SHALL (SHALL NOT): The words “shall” or “shall not” denote a requirement. “Shall” or “Shall not” is used whenever the objectively verifiable criterion for conformance is to be fulfilled from which no deviation is permitted. Its use shall not be avoided on the grounds that compliance with the standard is considered voluntary.

RECOMMENDATION – SHOULD (SHOULD NOT): The words “should” or “should not” denote a preferred recommendation. “Should” or “Should not” is used whenever noncompliance with the specific recommendation is permissible and a suggested possible choice or course of action is deemed to be particularly suitable without necessarily mentioning or excluding others. Its use shall not be substituted for “shall” or “shall not” on the grounds that compliance with the standard is considered voluntary.

PERMISSION – MAY (MAY NOT/NEED NOT): This phrasing is used to give consent or liberty (or opportunity) to do something.

POSSIBILITY AND CAPABILITY – CAN (CANNOT): This is used when covering the conditions:

- Expected or conceivable material, physical or causal outcome.
- Ability, fitness, or quality necessary to do or achieve a specified thing.

APPENDIX C - DEFINITIONS

Table 1 illustrates common terminology used within this document. The definitions presented are the intended meaning of the terms in the context of this document.

Table 1 – Definitions

Term	Definition
CDM	**C**ommon **D**ata **M**odel is the term used to describe the format of the published data from plant floor devices.
DeviceID	A DeviceID identifies each MQTT client that connects to an MQTT broker. Devices that publish and/or subscribe are both MQTT clients to the broker. The broker uses the DeviceID to identify the client and the current state of the client. Therefore, this ID is unique per client and broker. An empty DeviceID can be sent if the state does not need to be held by the broker. The empty DeviceID results in a connection without any state. In this case, the clean session flag must be set to true, or the broker rejects the connection.
IIoT	Industrial Internet of Things (IoT or IIoT) is a term for all the various sets of hardware pieces that work together through Internet of Things connectivity to help enhance manufacturing and industrial processes.
IIoT Asset	Any device or combination of devices to which IIoT data can be or is associated.
IIoT Platform	The IT Architecture and software used to ingest and visualize data from the devices and controllers on the production floor.
JSON	**J**ava **S**cript **O**bject **N**otation (JSON) is a standard open source data message format that is preferable due to its straightforward human readable format as well as its relative simplicity to generate from smart IIoT enabled devices. This file format uses human-readable text to transmit data objects consisting of attribute-value pairs and array data types. It is a common data format used for asynchronous browser-server communication, and as a replacement for XML in some AJAX-style systems.
LWT	**L**ast **W**ill and **T**estament is a feature used between an MQTT client and the MQTT broker when a client disconnects improperly or unexpectedly drops without sending a proper disconnect message. If used, a LWT message is stored on the Broker and sends this message to all clients that subscribe to it in the event of an unexpected disconnect.
MQTT	**M**essage **Q**ueuing **T**elemetry **T**ransport (MQTT) - MQTT is an ISO standard (ISO/IEC PRF 20922) Internet of Things publish-subscribe-based messaging protocol that decouples producers and consumers of M2M (machine to machine) device data. It works on top of the TCP/IP protocol and is designed for connections with remote locations where a "small code footprint" is required or the network bandwidth is limited. The MQTT publish-subscribe messaging pattern requires a message broker.
MQTT Topic	MQTT Topics are structured in a series of levels similar to folders and files in a file system. The broker uses “topics” to filter messages for each connected client. The topic consists of one or more topic levels. Each topic level is separated by a forward slash (topic level separator) as a delimiter.
OEM	Original Equipment Manufacturer
QoS Levels	**Q**uality **o**f **S**ervice (QoS) Level is an agreement between the sender of a message and the receiver of a message that defines the guarantee of delivery for a specific message. All message types use QoS 1 unless specified differently.

Term	Definition
	There are three QoS levels in MQTT: **QoS 0** – send at most once. This service level guarantees a best-effort delivery. There is no guarantee of delivery. The recipient does not acknowledge receipt of the message and the message is not stored or re-transmitted by the sender. QoS level 0 is often called “fire and forget” and provides the same guarantee as the underlying TCP protocol. **QoS 1** – send at least once. QoS level 1 guarantees that a message is delivered at least one time to the receiver. The sender stores the message until it gets a PUBACK packet from the receiver that acknowledges receipt of the message. It is possible for a message to be sent or delivered multiple times. **QoS 2** – send exactly once. This is the highest level of service in MQTT. This level guarantees that each message is received only once by the intended recipients. QoS 2 is the safest and slowest quality of service level. The guarantee is provided by at least two request/response flows (a four-part handshake) between the sender and the receiver.
Schema Philosophy	Schemas follow a major.minor-patch version philosophy. **Major version** – A Major version is incremented with a change that breaks backward compatibility such as a change in data format, tighter value restrictions, deleting keys, or values from an enumeration list or required parameter setting. **Minor version** – A Minor version is incremented with a change that maintains backward compatibility such as the addition of key:value pairs, loosening value restrictions, or adding values to enumeration lists. **Patch version** – A Patch version is incremented with changes that are bug fixes, descriptions updates, or other changes that are done in a controlled manner to not risk message validation issues.
Smart Manufacturing	A universal methodology for data communication, machine language, and reporting to and from manufacturing devices in the production environment.
Transaction Counter	The IIoT Platform stores the transaction counter when a message is successfully processed. The counter received should be one number greater than the counter stored. If the difference is greater than one, that means that messages were lost. The transaction counter is tracked separately for each message type in each source.
Waveform	**Domain** – independent parameter of a waveform against which the range is observed. **Range** – observed value of the measurement within a specified domain.

APPENDIX D - REFERENCES

For dated references, only the edition cited applies. For undated references, the latest published edition of the referenced document (including any attachments) applies.

Table 2 – References

Reference Title / Description	Website/URL	Hyperlink
ISA95, Enterprise-Control System Integration	https://www.isa.org/isa95/	Website
ISO/IEC Directives Part 2 Clause 7 Verbal forms for expression of provisions	https://www.iec.ch/members_experts/refdocs/iec/isoiecdir-2%7Bed7.0%7Den.pdf	Website
JSON.org Homepage	https://www.json.org/	Website
JSON-schema.org documentation on object properties	https://json-schema.org/understanding-json-schema/reference/object.html	Website
JSON-schema.org documentation on pattern expressions	https://json-schema.org/understanding-json-schema/reference/regular_expressions.html	Website
MQTT.org Homepage	http://mqtt.org/	Website

APPENDIX E - COMMON DATA MODEL (CDM) AND MESSAGE DELIVERY OVERVIEW

E.1 COMMON DATA MODEL (CDM)

Figure 5 shows how CDM messages are delivered via Message Queuing Telemetry Transport (MQTT) protocol architecture. This enables multiple Enterprise systems to subscribe to and ingest various manufacturing data via a pub/sub system called MQTT. This allows devices to publish data to a broker and applications to subscribe to that data. Any plant floor device be it legacy or IIoT compliant communicates to an MQTT broker using JSON formatted payloads using the common structure detailed in this specification.

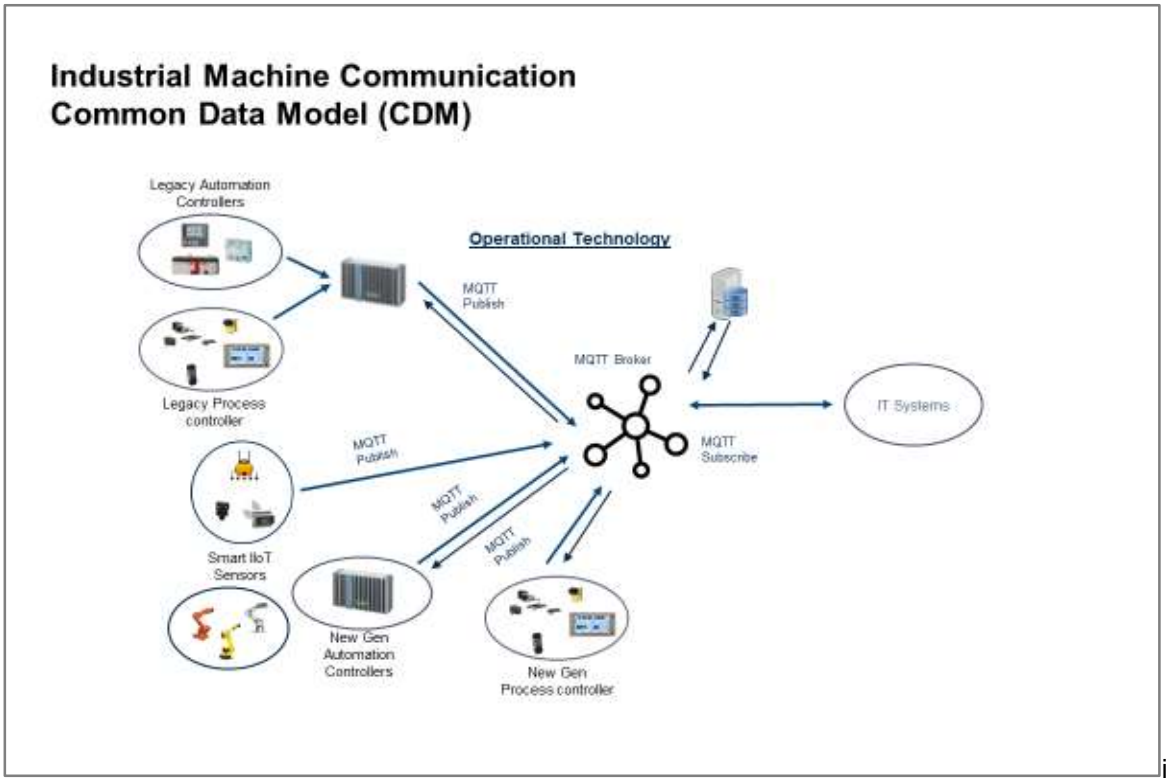

Figure 5 – Common Data Model (CDM)

E.2 MESSAGE QUEUING TELEMETRY TRANSPORT (MQTT) DEFINITION

Message Queuing Telemetry Transport (MQTT) is an ISO standard (ISO/IEC PRF 20922) Internet of Things (IoT) publish-subscribe-based messaging protocol that decouples producers and consumers of device data. It works on top of the TCP/IP protocol and is designed for connections with remote locations where a "small code footprint" is required or the network bandwidth is limited. The MQTT publish-subscribe messaging pattern requires a message broker.

E.3 PAYLOAD DEFINITION

The payload is formatted in Java Script Object Notation (JSON). JSON is a standard open source data message format that is preferable due to its straightforward human readable format as well as its relative simplicity to generate from smart IIoT enabled devices. This file format uses human-readable text to transmit data objects consisting of key-value pairs and array data types.

E.4 INDUSTRIAL ASSETS

All IIoT Asset types publish data in a JSON encoded payload to an MQTT broker, directly or via an edge gateway for legacy applications.

The communication protocol between plant floor devices sending machine data to the collection system utilizes an MQTT transport via TCP/IP.

All PLC and plant floor devices that publish messages are time synced (e.g., Network Time Protocol (NTP), Secure NTP, and Precision Time Protocol (PTP)) to the central timeserver available to the manufacturing plant (facility).

In addition to the Plant floor device sending time stamped data, a time stamp of the data insertion into the database is also stored.

APPENDIX F - TABLES AND FIGURES

F.1 FIGURES

F.2 TABLES

APPENDIX G- REVISIONS

This document may be revised periodically on a prescribed frequency such as annually, or whenever a major change in the specification is determined by the governing party. *Table 3* contains the DATE of approval or effectivity, REVISION #, the affected SECTION, a brief SUMMARY OF CHANGES MADE, and NOTES.

Table 3 – Revision Summary

DATE	REVISION #	SECTION	SUMMARY OF CHANGES MADE	NOTES
2023-07-07	0	All	Initial Release	All

NOTICE ABOUT INTERIM REVISIONS

Editorial updates or clarifications may be made as “interim revisions.” Interim revisions are documented as “revision letters” and are available on the website at [3].