Machinery Information Base Data Structure

Rainer Beudert
Marketing Director
Schneider Electric

Ludwig Leurs
Ethernet Convergence
Bosch-Rexroth

Steve Zuponcic
Technology Manager
Rockwell Automation

Presented at the ODVA
2015 Industry Conference & 17th Annual Meeting
October 13-15, 2015
Frisco, Texas, USA

Abstract

The Special Interest Group (SIG) for Machinery Information seeks to optimize the integration of manufacturing machines within the industrial ecosystem. The SIG is developing guidelines for exchange of information between machines and supervisory systems. A basic machinery Information description and a data model to organize the base machinery information for exchange with upper level systems was created. An extended data structure that allows the mapping of other data model (e.g. ODVA's CIP Energy model) has been proposed for addition to the base machinery data model.

This paper will discuss the basic data structures that have been defined for machine to supervisory information exchange as well as the extended data models that convey additional information about the machinery or the process. The data is given context in terms of state and status of the machine. The types of machines and supervisory systems for which the data models, states, and statuses are intended are outlined. These data structures will be mapped to CIP, Sercos, and OPC UA to meet the needs of the industrial market and to make conversion of inter-network data transfers easier.

1. Introduction
Machinery is central to the production process and manufacturers strive to optimize how it integrates with other machines and supervisory systems in their business enterprise. The Internet of Things is exposing many new opportunities to extract value from machines and processes through revealing valuable information from otherwise stranded data from sensors and actuators. Analyzing that new data helps in making better decisions. Manufacturers traditionally benchmark their high-value equipment by asset turnover, which measures how efficiently a company's assets generate revenue. In today's economy, however, production professionals and business executives need to have a more holistic focus. In addition to asset (i.e., machinery) turnover, manufacturers must measure the ability of assets to help the enterprise meet overall business goals and adapt to rapidly changing market demands. Ease of integration is an important element of this equation.

ODVA supports optimizing machinery integration using an open and interoperable framework for communication which is comprehensive, scalable, secure, open and inclusive for both manufacturers and machine builders. To that end, ODVA has introduced the Optimization of Machine Integration initiative to define this framework and has engaged both OPC Foundation and Sercos International to foster cross-collaboration and innovation.

For machine builders, Optimization of Machine Integration (OMI™) provides opportunities for creating additional value through simplified communication between machines and from machines to supervisory systems such as SCADA and MES. By transforming data into information, OMI will provide tools for dynamic decision-making, thus maximizing machine productivity, improving machine performance and enhancing the preventive maintenance of machinery assets. As a result, OMI will create more value from machines, extend machinery life cycles, and will emerge as a natural sweet spot to help manufacturers meet their overall business objectives, including workforce, profitability and sustainability goals.

The purpose of this document is to give an update and preview of the work that ODVA's Machinery Information SIGSIG's is doing on OMI. This document defines a data model and data structure for machine-to-supervisory system interaction. This data model will be mapped to CIP, Sercos, and OPC-UA in order to meet the needs of the industrial market and to make conversion of inter-network data transfers easier.

2. Overview of the machinery information model

The Machinery Information SIG workplan defines two basic use cases driving the need for OMI and standardized data models as an ODVA deliverable. The first use case is for machine to machine communications and the second is for machine to supervisory communications. Of these two cases, machine to supervisory communications was prioritized as the starting point for this standards work and, hence, this whitepaper is focused solely on that topic. Machine to machine communications will be defined later.

2.1. Architectural framework, requirements and specific use cases

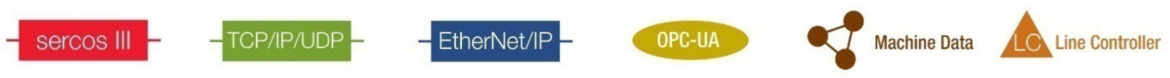
In an environment characterized by multiple networks and communication technologies, a key machine integration problem is how to streamline data transfer across heterogeneous communication interfaces. One driving motivation is to enable standard reporting methods and tools that would aid in the management of machines and the monitoring of their states. Without such reporting methods and tools, manufacturers must rely on customized, and often proprietary, solutions in order to exchange machine information across systems or transmit data back and forth with the machine. In the past, user groups have proposed some partial solutions to this problem, such as PackML and GEM.

The user-driven solutions do not always include standard definitions for how to put the data on a network in a consistent and seamless manner. An opportunity exists for a vendor-driven effort to complement the user-driven efforts to develop these standard definitions for the use cases for the integration of machines as described above. To take advantage of this opportunity, a protocol-neutral data model is needed that allows for flexibility in the propagation of machine information across multiple platforms that may or may not be on the same network.

Figure 1 represents an architectural example for machine-to-supervisory communication. In this illustration there are four coordinated machines composed of two machine pairs, plus two independent machines. The machines are part of one production line and the coordinated machine pairs are managed by a single line controller. In this case, the first machine pair is connected to the line controller over standard Ethernet, (**green line.**) The second pair is connected via EtherNet/IP, (**blue line**.) The line controller (LC) can also play a role in the transfer of data between machines and supervisory systems (e.g., MES). In each of these scenarios, the ability of each machine pair to communicate between themselves remains possible. A number of methods may be applied to transfer data between the machine level and the MES levels:

- Individual machines can exchange data the directly with the MES; or,
- machines can communicate to the MES via the line controller (LC). In this case the LC aggregates data from multiple subordinated machines and exchanges relevant data with the MES. The LC can either be a separate physical component in the architecture, as depicted in Figure 1, or it may be an embedded function within one of the machine sections.

The captions for each of the four scenarios describe the relevant data exchange based on the architecture depicted in accompanying figure.



Legend

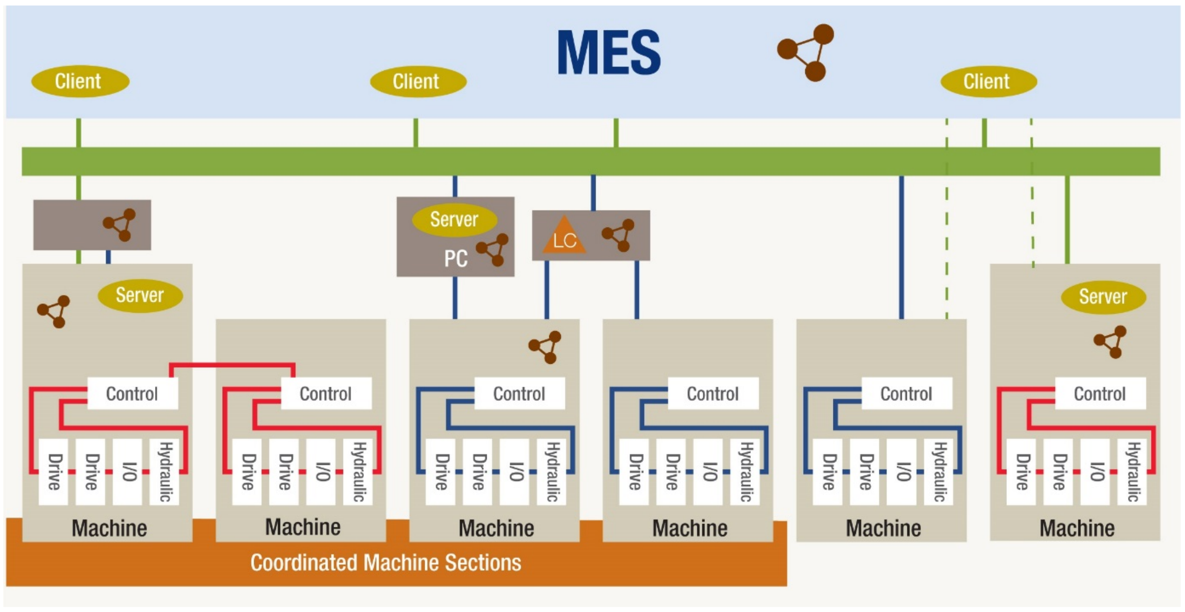


Figure 1 – Machine to Supervisory Communication with Separate Line Controller

Figure 1 depicts machine-to-supervisory communication with machine data that is communicated to the MES layer following various paths using an OPC UA server over standard TCP/IP or a native industrial communication protocol like EtherNet/IP. In this scenario, the external line controller can also serve as a data aggregator to the MES layer while also leveraging the data for its own use. The line controller can either be a separate physical component in the architecture, as depicted in Figure 1, or it may be an embedded function within one of the machine sections.

3. Base data structure

The ODVA machinery information data model consists of two parts. The first is the base machinery information data structure that is described in this section. The second is the extensible data structures for disciplines such as energy and condition monitoring as described in section 4.

The base machinery information data structure is foundational to the extensible data structures and is core to all data that would be communicated to the upper level systems. It contains all identifying information for the machine and is an important, integral, component of the organizational framework for every extended data structure that would be combined with it.

3.1. Description

The base machinery information data structure defines the information that is exchanged between a machine and another system to identify the core operational and informational capabilities of the machine. The base machinery data is the combination of the machine name plate data and the mandatory set of common information that helps define the specific machine.

3.2. Sources of data (machine types)

ODVA has identified the following list of machine types as the primary targets for application of this model. All machine types are candidates, and Table A below is a representative sample of machinery used in the manufacturing space and their North American Industry Classification System (NAICS) code numbers[1].

Table A – Machinery Types

Machinery classes	**NAICS Code**
Food Product Machinery Manufacturing	333294
Packaging Machinery Manufacturing	333993
Plastics and Rubber Industry Machinery Manufacturing	333220
Machine Tool (Metal Cutting Types) Manufacturing	333512
Oil and Gas Field Machinery and Equipment Manufacturing	333132
Engine, Turbine, and Power Transmission Equipment Manufacturing	33361
Conveyor and Conveying Equipment Manufacturing	333922
Paper Industry Machinery Manufacturing	333291
Semiconductor Machinery Manufacturing	333295
Mining Machinery and Equipment Manufacturing	333131
Printing Machinery and Equipment Manufacturing	333293
Mounting and Handling Machines	not classified

1.1. Users of data (MES, supervisory systems, etc.)

Many Manufacturing Execution System (MES) or Manufacturing Operations Management (MOM) applications require basic information about the machinery and equipment from which they are gathering data. Each MES/MOM software system has its unique functionality and method for collecting, storing, and analyzing the data. Table B is a representative sample of the classes of MES/MOM or supervisory control applications that will use the base machinery data described by the ODVA model.

[1] http://www.census.gov/eos/www/naics/

Table B – Users of Data

Applications that use machinery data (MES/MOM)
Administration
Alarms/events
ANDON
Batch/recipe management
Condition monitoring
Data exchange
Energy management
Laboratory management
Machine data access (MDA)
Maintenance management
Material management
Order management
Performance management
Process data access (PDA)
Quality management
Time-series data (historian)
Tool management
Traceability/genealogy
Version management

3.3. Machine states and status

3.3.1. Machine State Model and Machine Status Reporting

In industrial automation, machine states can be used to determine machine availability, the need for manual intervention, and more diagnostic information from the machine – generally called base machine status. Several machine state models are defined in standards for different areas of the industry, for example PackML [1] for packaging, Weihenstephan for beverage [2], GEM for semiconductor, and MTConnect for machine tools [3]. All standards have either been derived from or mapped to the ANSI/ISA 88 [4] standard. . (Reference “ISA-88.00.02-2001 Batch Control Part 2: Data Structures and Guidelines for Languages “)

From the manufacturing control area, two different requirements for machine status can be distinguished:

1. A coarse status that needs only information about the production state and the availability of the machine. Controlling the machine using this model is not needed, and because of the simplifications, control might not be possible.
2. A detailed status that is able to report problems in the production process. Production is often specific to the industry branch so the appropriate data model is mapped from the branch standard into the base machine data as an extension. This model also enables control of the machine and is able to determine measures to bring the machine back to the desired state.

For the purpose of associating the proper context of delivered information to the MES system or other systems receiving information from the machine, only the coarse status information is required to supply the needed reference or context. For this reason, a consolidated status model that combines states of several common state machines into a single status output model has been developed. The basic machine state model depicted in

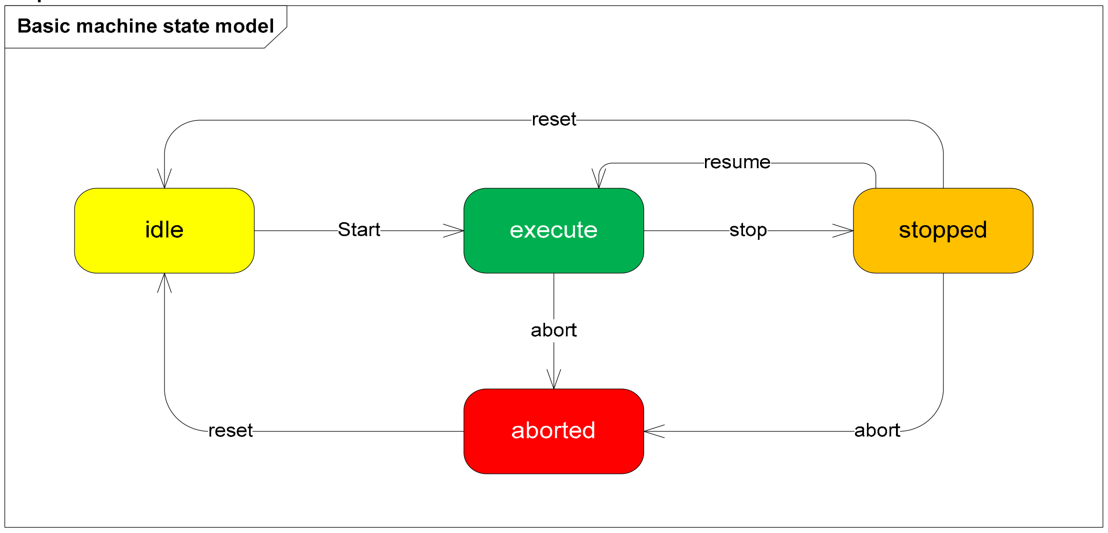


Figure 2: Basic Machine State Model

consists of four machine states, and four basic colors have been selected to represent the states for status reporting:

1. Idle: The machine is powered and ready to operate. There is no production running. The machine is ready to receive the “start” command to start production. The color selected for this state is yellow, because this state signals “ready to operate”.
2. Execute: The machine is processing material and producing the desired products. The color of this state is green, because production is running.
3. Stopped: The machine has either stopped or paused production. There can be a variety of reasons for this such as the completion of a production cycle, lack of material, product tailback, operator interaction, or others. As the effort to get the machine back to execute is considered relatively small, the color orange has been chosen for this state.
4. Aborted: The machine has stopped production because an error or an abort command occurred. Before the machine can go back to production, the error conditions have to be cleared and/or the abort command acknowledged. Because of the manual interaction necessary, the color red was chosen for this state.

The transitions between states in Figure 2 are for explanation only and not to be used for controlling the machine.

Different industries have standardized machine states for their specific use. The simple four color model unifies the reporting machine status.
Table C shows the mapping of PackML, Weihenstephan, GEM and MTConnect to ANSI/ISA 88. Transient states are considered to belong to the preceding steady state for status reporting purposes. The standards specific to dedicated industries define the states in detail as data type and values. This might be due to independent development of the standards or to special requirements for the industry specific MES.

Table C – Machine Status Reporting Cross Reference

ANSI/ISA 88	PackML	Weihenstephan	GEM	MTConnect	Base Machine Status
	Machine States				Machine Status
execute	execute (6)	operating (128)	executing	active	green (execute)
pausing					green (execute)
paused					orange (stopped)
idle	idle (4)	idle (32768)	idling	idle	yellow (idle)
stopped	stopped (2)	stopped (1)		stopped	orange (stopped)
starting	starting (3)	starting (2)	initializing		yellow (idle)
suspending	suspending (13)	-			green (execute)
suspended	suspended (5)	prepared (4)	pausing	interrupted	orange (stopped)
		lack (8)			
		tailback (16)			
		lack in branch line (32)			
		tailback in branch line (64)			
un-suspending	un-suspending (14)	-			orange (stopped)
stopping	stopping (7)	stopping			orange (stopped)
aborting	aborting (8)	aborting			red (aborted)
aborted	aborted (9)	equipment failure (1024)		emergency	red (aborted)
		external failure (2048)			
		emergency stop (4096)			
holding	holding (10)	holding (8192)			green (execute)
held	held (11)	held (16384)		feed hold	orange (stopped)
un-holding	un-holding (12)	-			orange (stopped)
completing	completing (16)	-			green (execute)
complete	complete (17)	-			orange (stopped)
resetting	resetting (15)	-			orange (stopped)
clearing	clearing (1)	-			red (aborted)
			setting up		

Note: It is intended that the Base Machine Status model will be a formal part of the ODVA specification as this construct becomes drafted into a CIPSE document.

1.1. Base data definitions

In an effort to promote a consistent means of communicating important data between machines and supervisory systems, a base data structure model has been developed. The data that is defined in the structure is intended for machine and equipment builders to organize relevant data in their machine control systems and for supervisory system creators and users to organize data in their applications. The Machinery Information SIG will create a CIP object definition that will make this data available as a standard, driving common data interfaces. The common definitions will help reduce the potential mismatch of data and the extra effort required to convert and translate data between systems.

1.1.1. Machine Data Flow

The data in the base machine structure consists of machinery configuration parameters or conditions, input data, and output data as shown in Figure 3. Various configuration parameters defined in Table D through Table G are exchanged between machinery and supervisory systems, typically during machine commissioning as the machine is being integrated with the plant supervisory system(s). Operational data required to setup or command the machinery is defined as input data in Table H and Table I. Similarly, operational data from the machinery that is relevant to the supervisory system(s) is defined as output data in Table J and Table K.

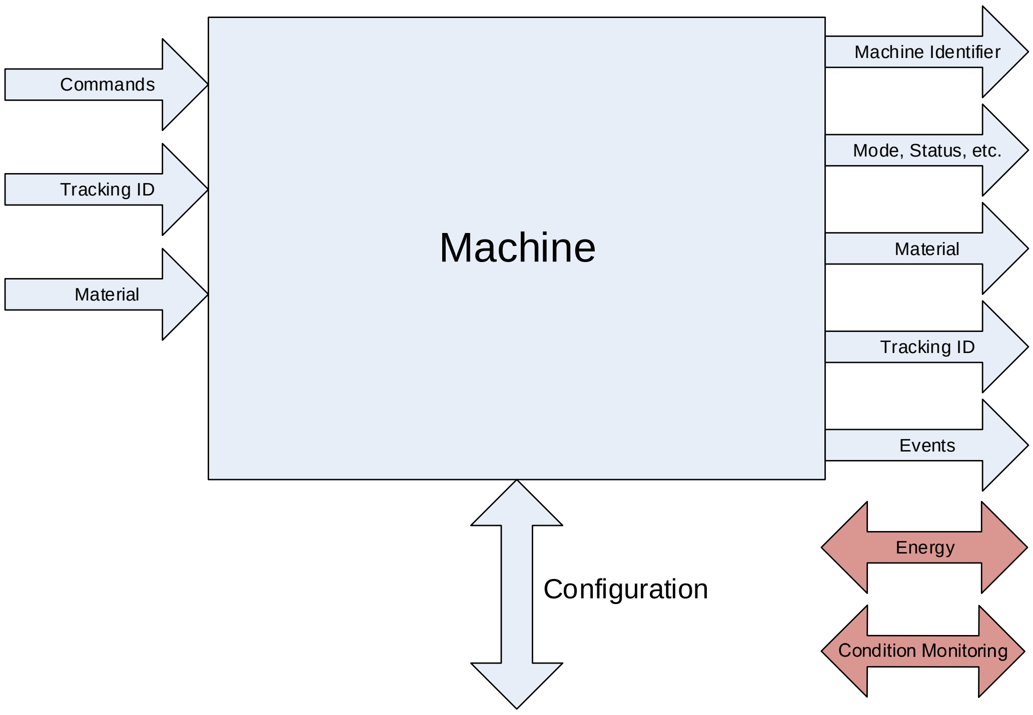


Figure 2 – Machine Data Flow

Table D – Machinery Configuration Data

This table defines all of the attributes required to describe fundamental identifying information of the machine

Attribute Name	Need in Implementation	Data Type	Description of Attribute	Semantics of Value
AssetID	Required	INT	End-User supplied unique ID string	The combination of the Asset ID and the Vendor Machine Serial Number form the Universally Unique Identifier (UUID). This represents the machine instance within the type.
EquipmentType	Required	STRING	End-User supplied machine type	Name of machine given by end-user per their standards.
MacnineVendorId	Required	STRING	End-User supplied unique ID string for the machine builder	Vendor ID number as used by the end-user.
EndUserDescription	Required	EUDescript[xx]	End-User information not otherwise defined	Relevant information the end user may assign to the machine.
VendorName	Required	STRING	Machine builder's name	Machine builder's name.
VendorModel	Required	STRING	Machine builder's model name for the machine	This represents the machine type.
VendorMachineSerialNumber	Required	STRING	Machine builder's unique ID string for the machine	The combination of the Asset ID and the Vendor Machine Serial Number form the Universally Unique Identifier (UUID). This represents the machine instance within the type.
VendorContact	Required	VendContact[xx]	Machine builder's address/contact info	Relevant information about the machine builder.
VendorInstallDate	Required	STRING	Machine builder's install date. Usually denotes the start date for the warranty period	Year, Month, Day giving the time of the warranty period.
VendorDescription	Optional	STRING[10]	Machine builder's description for what the machine does	Free form string describing in more detail the function of the machine.
Utilities	Required	Utilities[xx]	Machine builder's required utilities for machine to function	Designates the utilities (air, electricity, gas, etc.) required for the proper operation of the equipment.

Table E – Machinery Configuration Data – End User Description Structure

This table defines all of the attributes required to describe identifying information of the machine as required in the end user's facility and in the end user's MES or ERP systems.

Parameter Name	Need in Implementation	Data Type	Description of Parameter	Semantics of Value
MachineLocation	Required	STRING(10)	This is a 10 dimension array that holds all the information for the location of a machine, where each element in the array is given a specific meaning Each dimension of the array is 32 characters long.	See below.
	Required	String(0)	Country	Name of country where machine is located.
	Required	String(1)	State/Province	Name of state/province where machine is located.
	Required	String(2)	City	Name of city where machine is located.
	Required	String(3)	Facility	Name of facility where machine is located.
	Required	String(4)	Level	Name of level where machine is located.
	Required	String(5)	Cell/Quadrant/Zone	Name of cell/quadrant/ zone where machine is located.
	Required	String(6)	Machine ID	Machine identifier.
	Required	String(7)	User Defined	End user custom identifying information.
	Required	String(8)	User Defined	End user custom identifying information.
	Required	String(9)	User Defined	End user custom identifying information.

Table F – Machinery Configuration Data – Vendor Contact Structure

This table defines all of the attributes required to provide the machine vendor's contact information.

Parameter Name	Need in Implementation	Data Type	Description of Parameter	Semantics of Value
VendorContactInformation	Required	STRING(10)	This is a 10 dimension array that holds all the information for the location of a machine, where each element in the array is given a specific meaning Each dimension of the array is 32 characters long.	See below.
	Required	String(0)	Street number	Street number where vendor is located.
	Required	String(1)	Street Name	Name of Street where vendor is located.
	Required	String(2)	City	Name of City where vendor is located.
	Required	String(3)	State/Province	Name of State/Province vendor is located.
	Required	String(4)	Country	Name of Country where vendor is located.
	Required	String(5)	ZIP/postal Code	Name of ZIP/postal code where vendor is located.
	Required	String(6)	Contact Name	Name of person to contact at vendor location.
	Required	String(7)	Phone number	Phone number for vendor location.
	Required	String(8)	Phone extension	Phone extension at vendor location.
	Required	String(9)	Email address	Email address of contact at vendor location.

Table G – Machinery Configuration Data – Utilities Structure

This table is meant to define the attributes associated with describing all of the energy sources and associated utilities that are required for the machine to operate properly.

Parameter Name	Need in Implementation	Data Type	Description of Parameter	Semantics of Value
ID	Required	INTEGER	ID value for type of utility	Designates the ID of the utility as standardized by the end user. (air, gas, electricity, coolant, etc.).
Name	Required	STRING	Utility name	Actual name.
Value	Required	FLOAT	Utility set point	Design operating setpoint for this utility in order to achieve proper machine operation.
Units	Required	STRING	Utility units	Engineering units for the utility.
ScalerValue	Required	FLOAT	Utility scaling value	Any scaling value that should be applied given there may be a conversion or transformation of the utility with respect to some normalized reference.
ScalerUnits	Required	STRING	Utility scaled value units	Engineering units of the utility after a given transformation.

Table H – Machinery Input Data

This table defines the attributes associated with any fundamental operating inputs to the machine

Parameter Name	Need in Implementation	Data Type	Description of Parameter	Semantics of Value
State	Required	DINT	State change command: e.g., PackML Model	State of the machine as defined in state model that is being used (see **Fehler! Kein gültiges Resultat für Tabelle.** for examples).
CommandRate	Required	FLOAT	Production rate of the machine	Target rate at which the process should run.
CommandRateUnit	Required	STRING	Units of production	Units to apply to the command and production rate of the machine.
TrackingID	Required	STRING	Current process ID used for tracking production	Identifier for the produced parts (or material). In combination with the serial number and production time, material can be traced back to the raw material and production conditions.
ResetCounts	Required	BIT	Set all material total counts to zero	Resets the count for the produced material.
Material	Required	Material_In[xx]	Material setpoints	Array of material structure to describe all material used in the machine as input to production.

Table I – Machinery Input – Material Input Data Structure

This table defines the attributes associated with tracking the material that is fed into the machine

Parameter Name	Need in Implementation	Data Type	Description of Parameter	Semantics of Value
ID	Required	STRING	ID value for material	ID for the Input Material of the production machine.
Name	Required	STRING	Material name	Name of the input material.
TrackingID	Required	STRING	Current process ID used for tracking material	Tracking ID of the material to trace back to the production of this material.

Parameter Name	Need in Implementation	Data Type	Description of Parameter	Semantics of Value
Scaler	Required	FLOAT	Ratio of command rate to material rate	Scaling factor for material input to calculate transformation of material to finished product.

Table J – Machine Output Data

This table defines the attributes associated with the fundamental outputs of the machine.

Parameter Name	Need in Implementation	Data Type	Description of Parameter	Semantics of Value
RemoteEnabled	Required	BIT	Remote state and process rate commands are allowed	Reports whether or not the machine can be controlled from a remote location (i.e., not in the immediate vicinity of the machine such as from a central control room).
Mode	Required	DINT	Current active mode	Operating mode of the machine as defined by the machine/equipment builder and/or the machinery standard (e.g., PackML).
State	Required	DINT	Current active state: e.g., PackML Model	Operating state of the machine as defined by the machine/equipment builder and/or the machinery standard (e.g., PackML).
Status	Required	DINT	Current active status: ODVA Model (green, yellow, orange, red)	One of the four designated statuses (see **Fehler! Kein gültiges Resultat für Tabelle.**: enumeration 1=Green, 2=Yellow, 3=Orange, 4=Red).
TrackingID	Required	STRING	Current process ID used for tracking production	Unique identifier assigned by the end user business process to track the product during production.
Consumed	Optional	Material_Out[xx]	Material consumed as an input to the process	Characteristics of materials consumed by the production process; can be applied or nulled.
InProcess	Optional	Material_Out[xx]	Material currently being processed	Characteristics of production work in process; can be applied or nulled.
Produced	Optional	Material_Out[xx]	Material produced by the process	Characteristics of good products produced; can be applied or nulled.

Parameter Name	Need in Implementation	Data Type	Description of Parameter	Semantics of Value
Waste	Optional	Material_Out[xx]	Material lost due to reject or defects in the process	Characteristics of waste or scrap products produced; can be applied or nulled.
EventActive	Required	BIT	Event active indicator	Flags that a machine event is occurring (used together with Event Description and Event ID).
EventDescription	Optional	STRING	Event description	Literal description of the event in process when the Event Active flag is set.
EventID	Optional	DINT	Numeric value used to lookup description if one is not provided	References a specific string in a lookup table to describe the event in process when the Event Active flag is set if Event Description is not used.
ResetCountsDone	Required	BIT	Material totals have been set to zero	Acknowledge that product tracking counts have been reset after a ResetCounts action has been initiated.

Table K – Material Output Data Structure

This table defines the attributes associated with tracking the material or product that is output from the machine

Parameter Name	Need in Implementation	Data Type	Description of Parameter	Semantics of Value
ID	Required	STRING	ID value for material	Unique identifier for the material.
Name	Required	STRING	Material name	Text description of the material.
TrackingID	Required	STRING	Current process ID used for tracking material	Tracking identifier for the material in the relevant stage of the production process.
Available	Required	BOOL	Material flow is available	Status indication of the material flow as defined by the machine/equipment builder and/or the machinery standard (e.g., PackML).
IdealRate	Required	FLOAT	Ideal material usage rate based on command rate and scaler	Ideal rate at which the material will be consumed in the process.
RawTotal	Required	FLOAT	Material usage before scaling	Actual raw count of product that has been produced in the current production run.

Parameter Name	Need in Implementation	Data Type	Description of Parameter	Semantics of Value
Scaler	Required	FLOAT	Ratio of command rate to material rate	Scaling factor for material usage to calculate transformation of material to finished product.
Total	Required	FLOAT	Total material usage after scaling	Scaled material usage total for use by external systems.

4. Extensible data structures

Extensible data structures are structures that can be added to the base data structure from a predefined data set to offer a very flexible way to fulfill varying and future requirements. Some reasons to introduce extensible data types are to:

- Enable low cost or constrained devices and machines to offer full information support. This works by limiting the information to be stored in the device or machine to the mandatory data set. Other data may be stored outside the device and referenced by a link or similar construct.
- Eliminate inconsistent copies of data. Common data can be stored in a single location outside the device.
- Add new data structures to the device, machinery or system at any time after deployment to take advantage of new and emerging applications and technologies without changing the common data.

4.1. Static and dynamic data

Static data does not change over time and therefore can be stored outside the device. The minimum data set would be the identification data set that enables any supervisory system to uniquely identify the device. All other information could be stored outside the device and represented by a link or lookup.

Dynamic data are subject to change over time. This makes it likely that this type of data will be stored in the device. The minimum set of dynamic data needs to be stored locally. Additional data can be store remotely. The mandatory set of dynamic data (Common) should be stored locally.

4.2. Identification

Each object (structure) needs a unique identifier. In the case of a machine, the Universally Unique Identifier (UUID) would be the combination of the Vendor Machine Serial Number and the Asset ID (assigned by the end user).

4.3. Linked Information

Linked Information could reside on a server in, for example, an XML data file (e.g., //Server/dir/machine/uuid/ODVA.desc.xml). Relative path names and permalinks are permitted.

4.4. Optional features

Features of a machine are listed in the mandatory set of machine information. Only supported features are listed. Predefined features can reference a publicly available description that could be available worldwide on a public server (e.g., B2MML or JavaScript styles).

Special features reference a local description or a machine vendor/user specific description optionally residing outside the machine.

All optional feature descriptions need to have:

- a unique identifier
- description in an accessible data structure (e.g., XML file) either local or pointed to by link or lookup
- data that can be static/dynamic and local/remote

4.5. **XML description**

XML files contain syntax and semantics, but also pure text readable by humans, not by machines.

4.6. Machine Model

The machine model consists of a base machine object and additional (optional) objects. The optional objects can be predefined (e.g., as Energy object and Condition Monitoring Object as depicted in Figure 3) or independently defined. The base machine object defines the unique machine identification, the set of features, and the basic status/state model. Optionally it contains an extended state model and more information about the production process.

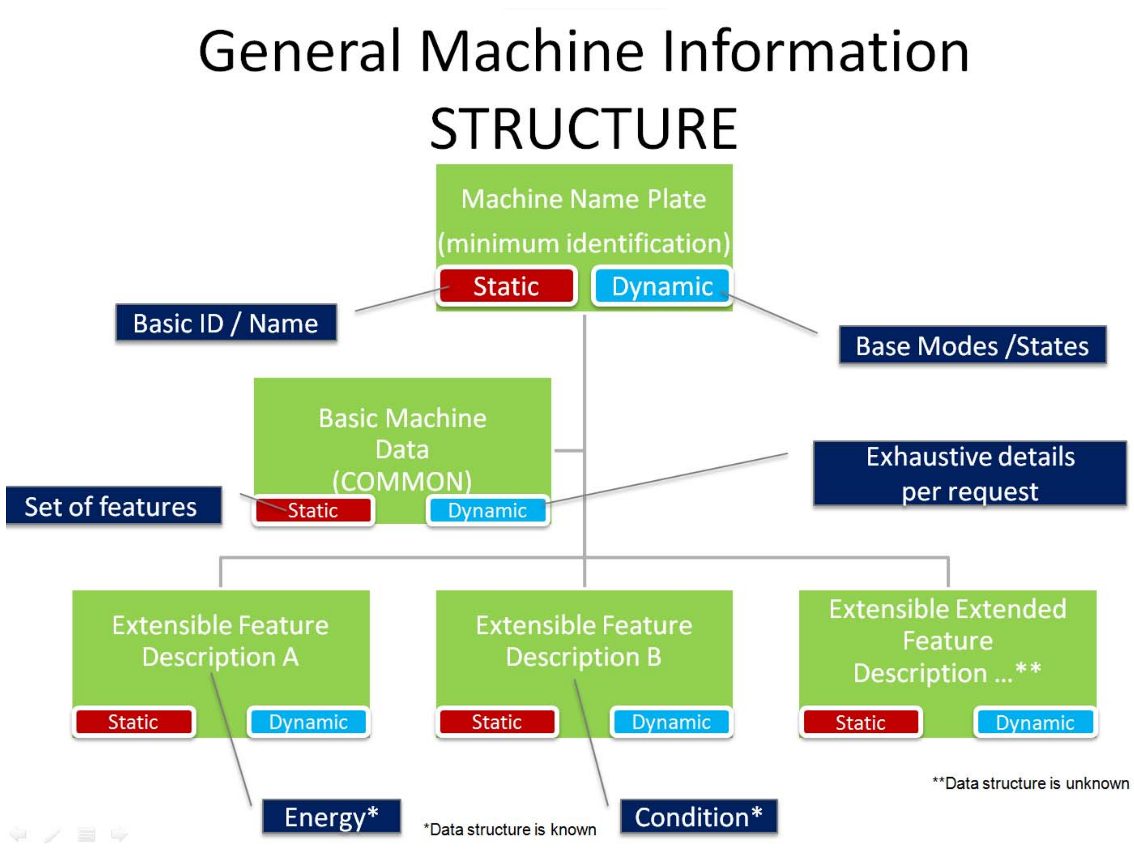

General Machine Information STRUCTURE
Machine Name Plate (minimum identification)
Static
Dynamic
Basic ID / Name
Base Modes /States
Basic Machine Data (COMMON)
Static
Dynamic
Set of features
Exhaustive details per request
Extensible Feature Description A
Static
Dynamic
Extensible Feature Description B
Static
Dynamic
Extensible Extended Feature Description ...**
Static
Dynamic
**Data structure is unknown
Energy*
*Data structure is known
Condition*

Figure 3 – General Machine Information Structure

5. Conclusion

The attainment of a standard data structure as mapped through CIP, Sercos, and OPC-UA will create many benefactors. The machine supplier is given a common “socket” for which standard “plugs” will interface. This allows the equipment designer to efficiently design equipment that has high value to his customer, making his equipment more attractive over competition. With a standard interface, there is no longer the need to design each machine differently for each customer. All data is mapped to known structures where pre-defined information can be readily collected.

The end-user benefits from a ubiquitous data model that easily traverses multiple protocols and networks, allowing for fluid bridging of data across different interfaces and mediums. The pre-defined data structures act like multi-pinned connectors that can be fitted into MES and ERP systems which are designed to accept the same standard.

This common data model enables the IoT promise of data enabled devices – where the end “device”, in this case, is a machine. In defining known structures and locations for machine information – all data is accessible for historization, asset management, supply chain management, manufacturing execution, and many other types of analytics and management tools.

The ODVA Machinery Information SIG will continue to drive out this capability to allow for true Optimization of Machine Integration.

References

[1] ISA, „Machine and Unit States:An Implementation Example of ISA88,“ ISA, Research Triangle Park, NC, 2008.

[2] ANSI/ISA, „Batch Control Part 1: Models and Terminology,“ ISA, Research Triangle Park, NC, 2010.

[3] A. Kather, C. Haufe und T. Voigt, „WS Pack Specification of the Interface Content (Part 2),“ Lehrstuhl für Lebemsmittelverpackungstechnik, TU München, München, 2014.

[4] J. Turner, „MTConnect Standard Part 3 - Streams, Events, Samples and Conditions V1.3.0,“ MTConnect Institute, 2014.