OMAC Packaging Working Group

PackML™ Committee

Tag Naming Technical Subteam (TST)

Tag Naming Guidelines

Version 2.0
26th Dec 2003

Please submit any comments to the PackML team via:

Larry Trunek, PackML tags subteam chair
Miller Brewing Company
3939 W. Highland Blvd.
Milwaukee, WI
USA
Phone: (414) 931-2953
Fax: (414) 931-6061
Email: ltrunek@mbco.com

The information contained herein is intended as a *guideline* for the control and automation of packaging machinery and systems by end users, equipment builders, and technology suppliers. This document is a *work-in-progress* and will be updated regularly, but no more frequently than three times per year following the general meetings of the Workgroup.

1. EXECUTIVE SUMMARY

Pack*Tags* are named data elements used for open architecture, interoperable data exchange in packaging machinery. In Pack*Tags* version 1.0, an initial set of Pack*Tags* sufficient for computing machine performance metrics was defined. In Pack*Tags* version 2.0, that initial set was expanded and adapted to support more of the PLCs that are in common use in the packaging industry today.

2. INTRODUCTION

Pack*ML*'s mission is to develop open architecture naming convention guidelines for communications between production machinery within the packaging industry. Three guidelines have been released to date, and are available on the OMAC.org and Pack*ML.org* websites. The Pack*ML* Line Type Definitions define the different levels of machine integration in common use today. The Pack*ML* State Model provides a uniform set of machine states, so that all packaging machinery can be looked at in a common way. Pack*ML* Machine Modes Definition document, defined automatic, semi-automatic and manual operations. This work addresses the third major objective of Pack*ML*, tag names and definitions of data sets. These include the fundamental names of the data elements (or "tags") as well as the data types, values, ranges and where necessary data structures – Pack*Tags*.

Consistent meanings of the data elements will be defined. Examples from the Pack*ML* State Model are the state names, such as "Producing", "Stopping", and "Aborted". Other examples are "total run time", "total productive time", "total downtime", "wastage", "efficiency", etc. From the existing work, and new Pack*Tags* that are suggested, we will arrive at a common set of well-defined naming conventions that the OMAC Packaging Workgroup (OPW) can recommend as guidelines for wide adoption in the packaging industry.

2.1 Plug-and-Pack

Tag name guidelines are useful for several reasons. Many of these fall under the broad set of goals that we call Plug-and-Pack™. This name is meant to suggest that it is to packaging machines what "plug and play" is to computing machines. Packaging users want their equipment to be easy to integrate, and Plug-and-Pack represents the ideal of machinery that is extraordinarily easy to integrate. Plug-and-Pack also implies multivendor interoperability.

There is much more to Plug-and-Pack than tag naming. Other OPW teams, such as Pack*Connect*™ and Pack*Soft*™ are working on some of the other aspects, such as network architecture, performance, and languages. Pack*Tags* "ride above" these aspects – they are on a different level. Our intention is to define Pack*Tags* such that they can be used with any underlying network and communication protocol. However, in their implementation, network protocols will need to be used.

Packaging users cite the following as key business drivers for achieving Plug-and-Pack™:

- Lowering the cost and reducing time for machine integration
- Allowing automatic startup and shutdown of packaging lines
- Increasing machine uptime via more rapid troubleshooting and facilitating root cause analysis of problems that cause downtime.
- Reducing operator and engineering training costs by providing a more uniform interface to machines for operators and engineers
- Lowering the cost of validation of production systems for Good Manufacturing Practices and other FDA regulations

2.2 Name Spaces

Pack*Tags* address the "name space" problem – in order to communicate, different entities of any type need to have a common vocabulary. In the case of packaging machinery that is enabled with computers and networks, the name space problem is a matter of defining a specific, unambiguous, and easy to use set of names and data types – Pack*Tags*.

Another way to look at name spaces is via databases. If all the information in each machine is stored in a database, data in each machine's database can be readily compared only if the field names are consistent – i.e., they use the same Pack*Tags*.

2.3 Domain of use

As noted above, Pack*Tags* are useful for machine-to-machine (intermachine) communications; for example between a filler and a capper. They will also be useful for intramachine communications; for example between motion controllers and PLCs on a single machine. In addition, Pack*Tags* will be used to exchange information between machines and higher-level information systems like SCADA systems, plant databases, and enterprise information systems.

2.4 Performance Metrics

Our subteam wanted to tackle a small area that would have immediate value to end-users, could be released for comment quickly and allowed us to work on our process. We chose the broad area of machine performance metrics such as machine efficiency, availability, etc. However, within this broad area, we chose to concentrate on information that represented the raw data necessary to calculate machine efficiency and other performance metrics without actually specifying or endorsing a particular calculation for these metrics.

The goal is to define a set of Pack*Tags* that would allow users to derive (or compute) most of the performance metrics that are in wide use in the industry. These are extremely useful, for

- equipment maintenance, including predictive maintenance and troubleshooting
- machine performance evaluations, in relation to business objectives such as productivity, profitability, quality, etc.
- performance comparisons between different machines
- deciding if and when to replace or update equipment
- new equipment specification and acceptance criteria
- work load scheduling and management
- asset management
- provide for more systematic logging of machine operations

There are a wide variety of different machinery performance metrics in use in industry. Often, companies and even different plants within companies will use different performance metrics, and there are cases where they are proprietary. Therefore, PackML could not provide for the computation of all the performance metrics users might need. For this reason, as stated above, the initial work focused not on calculating specific performance metrics, but rather on providing sufficient raw data to allow the calculation of virtually any performance metric in use. Later work may provide for tags which correspond to some of the well-known standard performance metrics, such as those in DIN standard 8743[1].

3. PACKTAGS

3.1 Methods of Supplying Raw Data

For basic machine availability (efficiency) computations, the time spent in each of the machine modes and states is sufficient. Two methods, elapsed time or event driven, can be used to supply the raw data for this. Some users prefer one of these, and some prefer the other, so this guideline supports both.

[1] DIN 8743, Concepts associated with packaging machines; definitions for time, output and their indication

Elapsed time: The "elapsed time" method is to have the machine keep track of the elapsed time spent in all the machine modes and states. Upon request, these times can be supplied to another system, which can then compute performance metrics. For example, a basic measure of availability might be the time spent in the Producing state divided by the total elapsed time spent in all states. Advantages of this method are (1) the requisite information can be obtained by requests only (the requesting system does not have to be set up to receive asynchronous messages), and (2) less message traffic is needed than the "event" method. Disadvantages are that it requires a timekeeping function in the machine, and the stored elapsed times can be lost if the power to the machine is cut off. The cumulative time tags in Appendix "A" are sufficient for this method.

Event: The "event" method is to have the machine record each change of state and mode whenever they occur. The other system receives this message, notes the time, and keeps track of the time spent in each state. It's advantages and disadvantages are opposite to those given above. In addition, this method can record supplemental information with each state transition, such as detailed information on the reason for the state transition. Note that a primitive hard-wired version of this method without any network is possible. The **State_Transition** tag in Appendix "A" is the only tag needed for this method.

3.2 Communication Format

This guideline has been generated to support machines operating from all OMAC PackML line types. Communication methods of control platform manufacturers vary widely, but must be capable of communicating with other machines on the line as well as SCADA systems. Advanced data types such as structured data types, arrays, and text strings are optionally supported by systems that can support the structure.

Pack Tags are broken out into two types; Control and Information. Control data is defined as data required to interface between machines and line control for coordination. Information data is described as data collected by higher level systems for machine performance analysis.[2]

3.3 PackTag prefix

PackTags will often be used in information systems that have a wealth of other named tags, so each PackTag should be prefixed with an identifying string to distinguish them from other tags that may have the same name. Previous work on S88 compatibility[3] shows how necessary this is, and established a precedent of using "PML_" as the prefix string. This makes it easy for a user who is unfamiliar with PackTags to look up their definitions.

3.4 PackTags name strings

[2] Each grouping of data should be in a contiguous grouping of registers to optimise communications.

[3] PackML State Model S88 Software Compatibility document, PackML, 3/22/2002

Many factory information systems do not allow for spaces in tag names, the guideline uses the common practice of substituting underline characters for spaces between words. The first letter of each word is capitalized for readability.[4] The total string shall not exceed 20 characters.[5] Thus, the exact text strings that should be used as tag names should be as follows:

PML_Xxxxxx_Yyyyyyyyy

For example:

PML_Cur_Mode,
PML_Mode_Time,
PML_Cum_Time_Modes, etc.

3.5 Data Types, Units, and Ranges

Appendix "A" shows the data type for each tag. They are as follows:

- Byte – 8 bit, or 0 to 255 in unsigned decimal format
- Integer – 32 bit, or 0 to 4294967295 in unsigned decimal format
- Real – 32-bit IEEE 754 standard floating point format (maximum value of 16,777,215 without introducing error in the integer portion of the number)
- Binary – Bit pattern within registers
- Time – seconds - Integer
- String – null-terminated ASCII
- Structure – a collection of data types. (This data type is typically reserved for higher level processors.)

Note that in future work, we may need to use a more formal method of defining tags data types, structures, units, ranges, etc. XML has often been suggested as a good method for this, but we have avoided it to date for simplicity's sake and to avoid using any method which would be protocol or language specific.

Date/Time stamps are not supported within this guideline due to problems with synchronization between controllers for accurate SCADA data collection. Date/Time stamping will be provided by higher level systems, and not maintained within the control platform.

Where applicable, the units and ranges should be as indicated in the Appendix "A" table.

[4] While IEC 61131 is not case sensitive, to ensure inter-operability with all systems it is recommended that the mixed case format be adhered to.

[5] Tag names are kept here to a maximum length of 20 characters to allow us to use them in some of the older processors. This 20 characters limit includes the "PML_" prefix.

3.7 Tag Details

The following is a detailed description of each tag.

3.7.1 Current Mode

Tag Name: PML_Cur_Mode
Data Type: Byte
Tag Descriptor: Current Mode
Comments: The response is one of a restricted set of values that define the current mode. Values 1-3 are identical to S88 modes, for compatibility reasons. The modes are:

0	Undefined
1	Automatic
2	Semi-Automatic
3	Manual
4	Idle

3.7.2 Mode Time

Tag Name: PML_Mode_Time
Data Type: Time
Tag Descriptor: Time in Current Mode

Comments: The response is the elapsed time the machine has spent in that mode since it entered that mode.

3.7.3 Cumulative Time In Each Mode

Tag Name: PML_Cum_Time_Modes
Data Type: Time
Tag Descriptor: Cumulative Time In Each Mode

Comments: This item represents numerous tags, one for each mode. The response is the cumulative elapsed time the machine has spent in each mode since it's timers and counters were reset. For controllers that do support arrays, an array of times is supplied, the length of the array being the number of possible modes. For controllers that do not support arrays, individual tags will need to be configured. Examples of the tags would be:

PML_Cum_Time_Auto
PML_Cum_Time_Man

3.7.4 Current State

Tag Name: PML_Cur_State
Data Type: Byte
Tag Descriptor: Current State

Comments: The response is one of a restricted set of integers that define the current state. These are one of the following:

0	undefined
1	“Off”
2	“Stopped”
3	“Starting”
4	“Ready”
5	“Standby”
6	“Producing”
7	“Stopping”
8	“Aborting”
9	“Aborted”
10	“Holding”
11	“Held”

3.7.5 State Time

Tag Name: PML_State_Time
Data Type: Time
Tag Descriptor: State Time

Comments: The response is the cumulative elapsed time the machine has spent in that state since it entered that state.

3.7.6 Cumulative Time in Each State

Tag Name: PML_Cum_Time_States
Data Type: Time
Tag Descriptor: Cumulative Time In Each State

Comments: This item consists of numerous tags, one for each state. The response is the cumulative elapsed time the machine has spent in each state since it’s timers and counters were reset. For controllers that do support arrays, an array of times is supplied, the length of the array being the number of possible modes. For controllers that do not support arrays, individual tags will need to be configured. Examples of the tags would be;

PML_Cum_Time_Stop
PML_Cum_Time_Off

3.7.7 Sequence Number

Tag Name: PML_Seq_Number
Data Type: Integer
Tag Descriptor: Sequence Number

Comments: Sequence Number is a free running number used to help in the verification of state transitions. It should increment by one for each successive state transition message.

3.7.8 Reason Code

Tag Name: PML_Reason_Code
Data Type: Integer
Tag Descriptor: Reason Code

Comments: The PML_Reason_Code is an integer, which will correspond to a user/machine builder defined list of reasons for the state transition to have been made. This could include error codes that have been propagated from other sub systems. Once the state transition has been initiated, this code must be locked in until the next state transition.

A standardized reason code will be available in Appendix "B".

3.7.9 Reason Code Index

Tag Name: PML_Reason_Index
Data Type: Integer
Tag Descriptor: Reason Code Index

Comments: The PML_Reason_Index is an integer, which will correspond to a specific PML_Reason_Code, and represents multiple instances of the same error on a given machine.
An example would be an open guard door on a machine that has multiple guard doors. The PML_Reason_Code would be a value of 13. The PML_Reason_Index for that fault would be an integer value of one through the number of doors on the machine, representing the door number that was open.
As with the PML_Reason_Code, once the state transition has been initiated, this code must be locked in until the next state transition.

3.7.10 Reason Text (Optional)

Tag Name: PML_Reason_Text
Data Type: String
Tag Descriptor: Reason Text

Comments: The Reason Text is an optional text string that states the reason for the state transition. For example, a transition from the Producing state to the Standby state may have as a reason "Materials Runout" or "Downstream Queue Full". Also, the Reason might optionally indicate the initiator of the transition, as for example an "Operator Pushbutton" or "Supervisory System Command".

In the case of a transition to the Abort state or any other fault or problem, the Reason should contain sufficient information to construct OEE, a Loss Tree, and to perform Root Cause Analysis[6].

There must be a direct correlation between PML_Reason_Code and PML_Reason_Text.

3.7.11 Supplemental Reason Code

Tag Name: PML_Reason_Code_Supp
Data Type: Binary
Tag Descriptor: Supplemental Reason Code

Comments: This code supplies additional information while in a state. Unlike the PML_Reason_Code, which identifies, and locks in, the reason for the transition, the supplementary reason code provides additional information while in the state. An example may be a machine stops for a downstream backup. While the machine is stopped, the operator opens a guard door. The downstream backup is removed. The supplemental reason code should identify the guard door, as well as any other reason for the machine not being allowed to transition states. Every bit should have a corresponding reason in Appendix "B". These reasons shall be different than the PML_Mat_Ready tag. An index such as PML_Reason_Index is not provided.

3.7.12 Current Machine Speed

Tag Name: PML_Cur_Mach_Spd
Data Type: Real
Units: Primary packages/minute
Tag Descriptor: Current Machine Speed

[6] OEE (Overall Equipment Efficiency) Definitions, PackML web site, slides 6 and 26, http://www.packml.org/OEE_Definitions.ppt

Comments: The response is the current speed of the machine in primary packages per minute. Keeping units in primary packages, allows for easier control integration. The following example is for a bottle line running at balance line speed doing a change over on the fly with flexible packaging.

Machine	Using Pack Counts	Using Primary packages
Bulk Depalletizer	41.6666 (24 pack equiv)	1,000
Filler	1,000	1,000
Labeler	1,000	1,000
Packer	66.666 (15 packs)	1,000

3.7.13 Current Speed Selected

Tag Name: PML_Cur_Spd_Sel
Data Type: Integer
Tag Descriptor: Current Speed Selected

Comments: The response is the current selected discrete speed of the machine. This allows monitoring of machine speed set points and frequency of speed changes. Additional speeds may be added between 9 and 99.

0	Undefined
1	Jog
2	Prime
3	Pre-Lube
4	Maintenance
5	Slow
6	Medium
7	High
8	Surge
9	Tracking
99	Analog control only

3.7.14 Machine Design Speed

Tag Name: PML_Mach_Design_Spd
Data Type: Real
Units: Primary Packages/minute
Tag Descriptor: Machine Design Speed

Comments: The response is the maximum design speed of the machine in primary packages per minute for the package configuration being run. This speed is NOT the maximum speed as specified by the manufacturer, but rather the speed the machine is designed to run in it's installed environment. Note that in practice the maximum speed of the machine as used for efficiency calculations will be a function of how it is set up and what products it is producing.

3.7.15 Number of Products Processed

Tag Name: PML_Prod_Processed
Data Type: Integer
Units: Primary Packages
Tag Descriptor: Number Products Processed

Comments: The response is the cumulative number of primary packages processed since the machine's counters and timers have been reset. This number represents machine through put, which includes good, defective and re-workable product.

3.7.16 Number of Defective Products

Tag Name: PML_Defect_Prod
Data Type: Integer
Units: Primary packages
Tag Descriptor: Number Defective Products

Comments: The response is the cumulative number of defective (non re-workable) primary packages processed since the machine's counters and timers have been reset.

3.7.17 Number of Re-workable Products

Tag Name: PML_Rework_Prod
Data Type: Integer
Units: Primary packages
Tag Descriptor: Number Re-workable Products

Comments: The response is the cumulative number of re-workable primary packages processed since the machine's counters and timers have been reset. It is assumed that these products will be reprocessed through the machine.

3.7.18 Machine Cycle Count

Tag Name: PML_Mach_Cycle
Data Type: Integer
Units: none
Tag Descriptor: Machine Cycle Count

Comments: Indicates the number of complete cycles of the machine with or without product being processed. This is derived from some sensor and then transmitted across the communication network.

3.7.19 Materials Ready

Tag Name: PML_Mat_Ready
Data Type: Binary
Units: none
Tag Descriptor: Materials Ready

Comments: Indicates materials are ready for processing. Comprised of a series of bits within a register(s) with 1 equaling ready, 0 equaling not ready. Each bit represents a different material. Materials are defined as all consumables such as product, cartons, labels, utilities, and glue.

3.7.20 Materials Low

Tag Name: PML_Mat_Low
Data Type: Binary
Units: none
Tag Descriptor: Materials Low

Comments: Indicates materials are running low. Comprised of a series of bits within a register(s) with 1 equaling material at full supply, 0 equaling material is running low. Each bit represents different material. Materials are defined as all consumables such as product, cartons, labels, utilities, and glue. This tag would indicate when one of the "*Material Ready*" supplies was becoming low. De-activation of one of the Binary components would not shut the machine down, but simply be an alarm to the operator. Continued operation of the machine in a "*Material Low*" may eventually result in the machine continuing to consume material and the corresponding bit in the "*Material Ready*" tag to go low, which would shut down the machine. Certain "*Material Low*" bits may cause the machine to change speed awaiting an operator in an attempt to preserve constant motion. If one of the materials doesn't have a low level warning, the bit for that material within the "*Material Low*" would be set high manually and left. There should never be a condition where a bit within "Materials Low" is high (1) while the corresponding bit within "Material Ready" is low (0). For example:

MACHINE READY TO RUN OR RUNNING, BUT RAW MATERIAL AND CO2 SUPPLY IS RUNNING LOW	Raw Material	Container	CO2	Compressed Air	Lubrication Water	Container Closures	Undefined / Unused	Undefined / Unused	Undefined / Unused	Undefined / Unused
Materials Ready	1	1	1	1	1	1	1	1	1	1
Materials Low	0	1	0	1	1	1	1	1	1	1

3.7.21 Transition Trigger Report

Tag Name: PML_Trans_Trigger
Data Type: Structure
Units: none
Tag Descriptor: Transition Trigger Report

Comments: Transmission of each trigger. This will be time and date stamp entry to a new state. The format is <Date and Time>, <Event Name>.

External triggers have significant importance to the other systems, yet may not immediately cause a state transition.

The <Event Name> is one of a restricted set of strings that define the current state. These are one of the following:

"Materials_Ready_True"
"Materials_Ready_False"
and others yet to be defined.

3.7.22 Package Ratio

Tag Name: PML_Prod_Ratio
Data Type: Integer
Units: Primary packages / Package
Tag Descriptor: Count of primary packages per secondary package

Comments: Contains the quantity of primary packages per current package being produced.

3.7.23 Package Reset

Tag Name: PML_Reset
Data Type: Boolean – Input into control device
Tag Descriptor: Reset timers and counters

Comments: Resets all timers and counters to zero. Refer to section 4.

3.7.24 Control Command

Tag Name: PML_Cntrl_Cmd
Data Type: Integer
Tag Descriptor: Control Command

Comments: Instruction to drive State change.

0	Undefined
1	Prepare
2	Start
3	Stop
4	Hold
5	Abort
6	Home
7	E-Stop

4. COMMANDS

In addition to tags, there is at least one command that appears to be necessary, a command to reset timers and counters associated with the performance tags. (Depending on the method of communication, commands could be more difficult to implement, so their incorporation in the guideline should be considered carefully.) Refer to section 3.7.23.

5. MACHINE IDENTIFICATION

The above tags provide a standardized method for communicating machine status and performance. It does not provide a standardized naming convention for associating a specific tag to a specific machine. Due to the diversity of available control architecture naming conventions, combined with the variability of how the end user sub-divides their facilities, along with user preference, no standard architecture could be recommended. It is highly recommend that each end user identify "their" standard for machine extensions for PackTags and then communicate those extensions to the technology provider or vendor. The machine location extensions may be prefixes or suffixes depending upon what the end users platform supports, and user preference. Possible options may be;

Plant.line.machine.PML_tag
PML_tag.Plant.line.machine
PML_tag.plant.area.line.machine

6. CONCLUSIONS

We have defined the second release of tags for PackML. This set is adequate to begin implementations that will provide real value to users and machine builders, by providing an open and interoperable way for packaging machinery to communicate with other machinery on packaging lines, with supervisory systems, and with enterprise information systems.

7. FUTURE WORK / ISSUES

Key areas for future work are:

- Publication of a set of performance metrics expressed in terms of PackML tags. Some examples are machine efficiency, production efficiency, availability etc.
- Review of the methods for the retrieval of tag information across different networks
- Internationalization of terminology
- Definition of recipes for packaging

8. REFERENCES

PackML Integration Brief V2.0
PackML Line Types Document
PackML State Definition Document

Appendix A

CONTROL

Spec #	Prefix	Tag Name[7]	Tag Descriptor	Data Type	Units
3.7.1	PML_	Cur_Mode	Current Mode	Byte	
3.7.4	PML_	Cur_ State	Current State	Byte	
3.7.12	PML_	Cur_Mach_Spd	Current Machine Speed	Real	Primary packages/Min
3.7.19	PML_	Mat_Ready	Materials Ready	Binary	
3.7.20	PML_	Mat_Low	Materials Low	Binary	
3.7.21	PML_	Trans_Trigger	Transition Trigger	Structure	
3.7.23	PML_	Reset	Package Reset	Boolean	
3.7.24	PML_	Cntrl_Cmd	Control Command	Integer	

INFORMATION

Spec #	Prefix	Tag Name	Tag Descriptor	Data Type	Units
3.7.2	PML_	Mode_Time	Time in Current Mode	Time	Seconds
3.7.3	PML	Cum_Time_Modes	Cumulative Time In All Modes	Time	Seconds
3.7.5	PML_	State_Time	State Time	Time	Seconds
3.7.6	PML	Cum_Time_States	Cumulative Time In All States	Time	Seconds
3.7.7	PML_	Seq_Number	Sequence Number	Integer	
3.7.8	PML_	Reason_Code	Reason Code	Integer	
3.7.9	PML_	Reason_Index	Reason Index	Integer	
3.7.10	PML_	Reason_Text	Reason Text	String	
3.7.11	PML_	Reason_Code_Supp	Supplemental Reason Code	Binary	
3.7.13	PML_	Cur_Spd_Sel	Current Speed Selected	Integer	
3.7.14	PML_	Mach_Design_Spd	Machine Design Speed	Real	Primary packages/Min
3.7.15	PML_	Prod_Processed	Number Products Processed	Integer	Primary packages
3.7.16	PML_	Defect_Prod	Number Defective Products	Integer	Primary packages
3.7.17	PML_	Rework_Prod	Number Re-workable Products	Integer	Primary packages
3.7.18	PML_	Mach_Cycle	Machine Cycle	Integer	
3.7.22	PML_	Prod_Ratio	Product Ratio	Integer	Primary packages/Pack

[7] Tag names are kept here to a maximum length of 20 characters to allow us to use them in some of the older processors. This 20 character limit includes the "PML_" prefix.

Appendix B - Reason Codes

Grouping

Reason #	Reason Text
1 – 32	Machine Internal Reason - Safeties - OMAC Defined
33 – 64	Machine Internal Reason – Operator Actions - OMAC Defined
65 – 256	Machine Internal Reason – Internal Machine faults – Product related - OMAC Defined
257 - 512	Machine Internal Reason – Internal Machine faults – Machine related - OMAC Defined
513 – 999	Machine Internal Reason – General Information - OMAC Defined
1000 – 1999	Machine Internal Reason – Vendor Defined
2000 - 2499	Machine Upstream Process Reason – OMAC Defined
2500 - 2999	Machine Upstream Process Reason – Vendor Defined
3000 – 3499	Machine Downstream Process Reason – OMAC Defined
3500 - 3999	Machine Downstream Process Reason – Vendor Defined
4000 – 4499	Out Of Service – OMAC Defined
4500 - 4999	Out Of Service – Vendor Defined

Detail (From above grouping)

Reason #	Reason Text
0	Undefined
1	E-Stop pushed
2	Perimeter Protection Fault
3	Mains turned off
4	Safety Gate/Guard Door Open
5 - 32	Reserved for future OMAC defined safety codes
33	Cycle Stop Button Pushed
34	Start Button Pushed
35	Reset Button Pushed
36	Jog Mode Selected
37	Automatic Mode Selected
38	Manual Mode Selected
39	Semi-Automatic Mode Selected
40 - 64	Reserved for future OMAC defined operator action codes
65	Material Jam
66 - 256	Reserved for future OMAC defined internal material related codes
257	Machine Jam
258	Electrical Overload
259	Mechanical Overload
260	Drive Fault

261	Drive Failure
262	Servo Axis Fault
263	Servo Axis Failure
264	Communication Error
265	PLC Error Code
266	Vacuum
267	Air Pressure
268	Voltage
269	Temperature
270	Hydraulic Pressure
271	Hydraulic Level
272	Hydraulic Temperature
273 - 512	Reserved for future OMAC defined internal machine related codes
513	Counter Preset Reached
514	Product Selected
515	Local Slow Speed Requested
516	Local Medium Speed Requested
517	Local High Speed Requested
518	Local Surge Speed Requested
519	Remote Speed Requested
520	Drive Warning
521	Servo Warning
522 - 998	Reserved for future OMAC defined general information related codes
999	Catch All - Unidentified internal reason
1000 - 1999	Vendor defined area for machine internal items
2000	Infeed Not turned On
2001	Infeed Overload
2002	Low Prime Material
2003	High Prime Material
2004 - 2498	Reserved for future OMAC defined upstream related codes
2499	Catch All - Unidentified upstream reason
2500 - 2999	Vendor defined area for upstream items
3000	Discharge Not Turned On
3001	Discharge Overload
3002	Discharge Blocked reason
3003	Discharge Cycle Stop reason
3004	Discharge Immediate Stop reason
3005 – 3498	Reserved for future OMAC defined downstream related codes
3499	Catch All - Unidentified downstream reason
3500 - 3999	Vendor defined area for downstream items

4000	Line Not Scheduled
4001	Planned Maintenance
4002	Meals and Rest
4003	Meetings
4004	Training
4005	No Materials
4006	Remote Stop Requested
4007	Machine Not Selected
4008	Changeover
4009	Lubrication
4010	Product count preset reached
4011	Setup Selected
4012 – 4499	Reserved for future OMAC defined Out of Service related codes
4500 - 4999	Vendor defined area for Out of Service items