European Plastics and Rubber Machinery

EUROMAP 83	OPC UA interfaces for plastics and rubber machinery – General Type definitions

Release 1.01, 28 January 2019
75 pages

This EUROMAP recommendation was prepared by the Technical Commission of EUROMAP.

www.euromap.org

EUROMAP Technical Commission
c/o VDMA • FV KuG

Lyoner Str. 18
DE 60528 Frankfurt am Main

Phone + 49 (69) 6603-1833
Fax + 49 (69) 6603-2833
E-Mail:euromap@vdma.org

History

Date	Changes
11 September 2017	Published as Release Candidate 1.00
04 May 2018	Published as Release 1.00

28 January 2019	Published as Release 1.01.
	• Some descriptions have been clarified
	• *Classification* of temperature zones added to *TemperatureZoneType* (clause 14) and *TemperatureZoneCycleParametersType* (clause 16.15)
	• In the *MachineInformationType* (clause 5) the *Property DeviceClass* derived from the *DeviceType* is made mandatory.
	• The following types have been added (see clauses 18 – 24):
	  ○ *ObjectTypes:*
	  ▪ *IdentificationType*
	  ▪ *MonitoredParameterType*
	  ▪ *ControlledParameterType*
	  ▪ *ClosedLoopControlType*
	  ▪ *MaintenanceType*
	  ○ *EventTypes:*
	  ▪ *HelpOffNormalAlarmType*
	  ○ *DataTypes:*
	  ▪ *PIDParametersDataType*
	  ▪ *ActiveErrorDataType*
	  ▪ *MaintenanceStatusEnumeration*
	• Addition corrections in model files:
	  ○ 17.3.5 *ProductionDatasetStatusType*.*Load*: *ValueRank* of input argument *Components* changed to OneDimension (was Scalar)
	  ○ 17.4.4 *ProductionDatasetInformationType*: *DataType* of *NumCavities* changed to UInt32 (was String)
	  ○ Typing errors in descriptions corrected

Contents Page

1 Introduction

1.1 Scope and Application

For the communication between different machines, manufacturer independent information models are required. For plastics and rubber machinery, these information models are based on OPC UA, a communication framework developed and provided by the OPC Foundation. While OPC UA provides the technology for the transfer of information, the definition which information is transferred in which form is fixed in Companion Specifications.

This recommendation defines a Companion Specification for general information regarding plastics and rubber machines. The intention is that *ObjectTypes* which can be used for several machines and applications are defined only once. For specific applications (e.g. connection of injection moulding machines to MES), it is extended by specific Companion Specifications (e.g. EUROMAP 77).

NOTE: Not all machines will support all *ObjectTypes*, e.g. *TemperatureZoneType*, *MouldType*.

1.2 OPC Unified Architecture

OPC is the interoperability standard for the secure and reliable exchange of data in the industrial automation space and in other industries. It is platform independent and ensures the seamless flow of information among devices from multiple vendors. The OPC Foundation is responsible for the development and maintenance of this standard.

The OPC standard is a series of specifications developed by industry vendors, end-users and software developers. These specifications define the interface between Clients and Servers, as well as Servers and Servers, including access to real-time data, monitoring of alarms and events, access to historical data and other applications.

When the standard was first released in 1996, its purpose was to abstract PLC specific protocols (such as Modbus, Profibus, etc.) into a standardized interface allowing HMI/SCADA systems to interface with a “middle-man” who would convert generic-OPC read/write requests into device-specific requests and vice-versa. As a result, an entire cottage industry of products emerged allowing end-users to implement systems using best-of-breed products all seamlessly interacting via OPC.

Initially, the OPC standard was restricted to the Windows operating system. As such, the acronym OPC was borne from OLE (object linking and embedding) for Process Control. These specifications, which are now known as OPC Classic, have enjoyed widespread adoption across multiple industries, including manufacturing, building automation, oil and gas, renewable energy and utilities, among others.

With the introduction of service-oriented architectures in manufacturing systems came new challenges in security and data modelling. The OPC Foundation developed the OPC UA specifications to address these needs, and at the same time provided a feature-rich technology open-platform architecture that was future-proof, scalable and extensible.

These are just some of the reasons why so many members and other technology organizations (collaborations) are turning to OPC UA for their interoperability platform.

1.3 References

Short name	Title	Version
OPC UA Part 1	OPC Unified Architecture – Part 1: Overview	1.04
OPC UA Part 3	OPC Unified Architecture – Part 3: Address Space Model	1.04
OPC UA Part 4	OPC Unified Architecture – Part 4: Services	1.04
OPC UA Part 5	OPC Unified Architecture – Part 5: Information Model	1.04
OPC UA Part 6	OPC Unified Architecture – Part 6: Mappings	1.04
OPC UA Part 7	OPC Unified Architecture – Part 7: Profiles	1.04
OPC UA Part 8	OPC Unified Architecture – Part 8: Data Access	1.04
OPC UA Part 9	OPC Unified Architecture – Part 9: Alarms and Conditions	1.04
OPC UA Part 11	OPC Unified Architecture – Part 11: Historical Access	1.03
OPC UA Part 12	OPC Unified Architecture – Part 12: Discovery	1.03
OPC UA Part 100	OPC Unified Architecture – Part 100: OPC UA for Devices	1.01

1.4 Namespaces

1.4.1 Namespace and identifiers for EUROMAP77 Information Model

This clause defines the numeric identifiers for all the numeric *NodeIds* defined in this specification. The identifiers are specified in a CSV file with the following syntax:

```
<SymbolName>, <Identifier>, <NodeClass>
```

where the *SymbolName* is either the *BrowseName* of a *Type Node* or the *BrowsePath* for an *Instance Node* that appears in the specification and the *Identifier* is the numeric value for the *NodeId*.

The *BrowsePath* for an *Instance Node* is constructed by appending the *BrowseName* of the instance *Node* to the *BrowseName* for the containing instance or type. An underscore character is used to separate each *BrowseName* in the path. Let's take for example, the *MachineInformationType ObjectType Node* which has the *ControllerName Property*. The **Name** for the *ControllerName InstanceDeclaration* within the *MachineInformationType* declaration is: *MachineInformationType_ControllerName*.

The *NamespaceUri* for all *NodeIds* defined here is http://www.euromap.org/euromap83/

The CSV released with this version of the specification can be found here:

http://www.euromap.org/files/Opc_Ua.EUROMAP83.1_01.NodeId.csv

NOTE: The latest CSV that is compatible with this version of the specification can be found here:

http://www.euromap.org/files/Opc_Ua.EUROMAP83.NodeId.csv

A computer processable version of the complete Information Model defined in this specification is also provided. It follows the XML Information Model schema syntax defined in OPC UA Part 6.

The Information Model Schema released with this version of the specification can be found here:

http://www.euromap.org/files/Opc_Ua.EUROMAP83.1_01.NodeSet2.xml

NOTE: The latest Information Model schema that is compatible with this version of the specification can be found here:

http://www.euromap.org/files/Opc_Ua.EUROMAP83.NodeSet2.xml

1.4.2 Namespace Metadata

Table 1 defines the namespace metadata for this specification. The *Object* is used to provide version information for the namespace and an indication about static *Nodes*. Static *Nodes* are identical for all *Attributes* in all *Servers*, including the *Value Attribute*. See OPC UA Part 5 for more details.

The information is provided as *Object* of type *NamespaceMetadataType*. This *Object* is a component of the *Namespaces Object* that is part of the *Server Object*. The *NamespaceMetadataType ObjectType* and its *Properties* are defined in OPC UA Part 5.

The version information is also provided as part of the ModelTableEntry in the UANodeSet XML file. The UANodeSet XML schema is defined in OPC UA Part 6.

Table 1 – NamespaceMetadata Object for this Specification

Attribute	**Value**		
BrowseName	Euromap83_NamespaceMetadata		
References	**BrowseName**	**DataType**	**Value**
HasProperty	NamespaceUri	String	http://www.euromap.org/euromap83/
HasProperty	NamespaceVersion	String	1.01
HasProperty	NamespacePublicationDate	DateTime	2019-01-28 08:00:00
HasProperty	IsNamespaceSubset	Boolean	False
HasProperty	StaticNodeIdTypes	IdType[]	{Numeric}
HasProperty	StaticNumericNodeIdRange	NumericRange[]	Null
HasProperty	StaticStringNodeIdPattern	String	Null

1.4.3 Handling of OPC UA namespaces

Namespaces are used by OPC UA to create unique identifiers across different naming authorities. The *Attributes NodeId* and *BrowseName* are identifiers. A node in the UA *Address Space* is unambiguously identified using a *NodeId*. Unlike *NodeIds*, the *BrowseName* cannot be used to unambiguously identify a node. Different nodes may have the same *BrowseName*. They are used to build a browse path between two nodes or to define a standard *Property*.

Servers may often choose to use the same namespace for the *NodeId* and the *BrowseName*. However, if they want to provide a standard *Property*, its *BrowseName* shall have the namespace of the standards body although the namespace of the *NodeId* reflects something else, for example the *EngineeringUnits* property. All *NodeIds* of nodes not defined in this specification shall not use the standard namespaces.

Table 2 provides a list of mandatory namespaces used in a EUROMAP OPC UA *Server*.

Table 2 – Namespaces used in a EUROMAP 83 Server

Namespace	**Description**	**Use**
http://opcfoundation.org/UA/	Namespace for *NodeIds* and *BrowseNames* defined in the OPC UA specification. This namespace shall have namespace index 0.	Mandatory
Local Server URI	Namespace for nodes defined in the local server. This may include types and instances used in a device represented by the server. This namespace shall have namespace index 1.	Mandatory
http://opcfoundation.org/UA/DI/	Namespace for *NodeIds* and *BrowseNames* defined in OPC UA Part 100. The namespace index is server specific.	Mandatory
http://www.euromap.org/euromap83/	Namespace for *NodeIds* and *BrowseNames* defined in this specification. The namespace index is server specific.	Mandatory
Application specific EUROMAP specification	Namespace for *NodeIds* and *BrowseNames* defined in application specific specification (e.g. EUROMAP 77).	Optional
Vendor specific types and instances	A server may provide vendor specific types like types derived from *MachineStatusType* or vendor specific instances of devices in a vendor specific namespace.	Optional

2 Conventions for definitions

The general OPC UA conventions for the definitions in tables are defined in IEC 62541-5:2015. In addition, in EUROMAP 83 the abbreviations defined in Table 3 are used for the definitions of object types.

Table 3 – Abbreviations for Modelling Rule

Abbreviation	Description
O	Modelling Rule Optional
M	Modelling Rule Mandatory
OP	Modelling Rule OptionalPlaceholder
MP	Modelling Rule MandatoryPlaceholder
R	AccessLevel Readable
W	AccessLevel Writeable
RW	AccessLevel Readable and Writable

Example: "M, RW" means that the parameter is mandatory and readable and writeable by a client.

3 General requirements

3.1 NodeIds

The *NodeIds* of the *Types* defined in EUROMAP 83 are mandatory and fixed in the provided XML-file. This applies only to the *Types* themselves and not to the child elements. *NodeIds* of generated instances are not fixed and assigned by the individual OPC UA server. Therefore, the namespace of the *NodeIds* is the Local Server URI with namespace index 1 or a vendor specific namespace with vendor specific index. However, a *Server* shall persist the *NodeIds* of the generated *Nodes*, that is, it shall not generate new *NodeIds* when rebooting.

NOTE: It is not mandatory that the servers of all machines from one manufacturer use the same *NodeIds* for the individual instances, so a client cannot expect this. Also, a software update of a machine may lead to new *NodeIds*.

3.2 AnalogItemType

For EUROMAP 83, wherever *AnalogItemType* is used, the property *EngineeringUnits* is mandatory. The unit system described in OPC UA Part 8 (“Codes for Units of Measurement (Recommendation N°. 20)” published by the “United Nations Centre for Trade Facilitation and Electronic Business” (see UN/CEFACT)) shall be used.

3.3 EventNotifier

If events are supported, the root node of the specific interface (e.g. IMM_MES_Interface for EUROMAP 77) shall set the *SubscribeToEvents* flag in the *EventNotifier* attribute.
If event history is supported, the root node of the specific interface (e.g. IMM_MES_Interface for EUROMAP 77) shall set the *HistoryRead* flag in the *EventNotifier* attribute.

3.4 Severity of events

Wherever a *Severity* property is used (e.g. in *MESMessage* in 11.4, all events based on *BaseEventType*), the following classification shall be used:

Table 4 – Severity Classes

Range of Severity	Description
667 – 1000	Messages of high urgency (Error)
334 – 666	Messages of medium urgency (Warning)
1 – 333	Messages of low urgency (Information)

4 Container concept

Several objects can occur several times in the parent object (e.g. several moulds in one machine). For these, container objects are modelled. The benefit is that all instances are collected in one object so that changes can be easily recognized by using a Property *NodeVersion* which can be subscribed by clients. According to OPC UA, Part 3, the instances of the container objects shall also trigger a *GeneralModelChangeEvent*.

The following container types are defined:

- *UsersType*
- *MouldsType*
- *TemperatureZones*
- *PowerUnits*

Within these containers, the child elements have the modelling rule *OptionalPlaceholder* which allows to add and remove instances of the objects dynamically.

The *BrowseNames* of the child elements are created with numbers as suffixes starting with 1 (e.g. “Mould_1”, “Mould_2” …). It is allowed to use numbers with leading zeros, e.g. “Mould_001” for better sorting. It is not mandatory to use successive numers. That means there can be gaps and it is also allowed to have e.g. two instances with BrowseNames “Mould_042” and “Mould_099”.

The objects have an *IsPresent*-flag. This allows to have a fixed number of instances prepared (e.g. if the maximum number of child components is limited by the design of the machine) and to indicate if the component is physically present. With this there are two possibilities: Dynamic creation fo instances or fixed number of instances. However, a client shall always subscribe the *NodeVersion* and/or *ModelChangeEvents* of the container object to get informed about changes.

NOTE: As the child elements are created by the server, the namespace of the *BrowseNames* is the Local Server URI with namespace index 1 or a vendor specific namespace with vendor specific index.

4.1 UsersType

This *ObjectType* is a container for the user(s). It is formally defined in Table 5.

Table 5 – UsersType Definition

Attribute	Value				
BrowseName	UsersType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType* defined in OPC UA Part 5					
HasProperty	Variable	0:NodeVersion	String	PropertyType	M, R
HasComponent	Object	User_<Nr>		UserType	OP

When instances for users are created, the *BrowseNames* shall be "User_<Nr>" (starting with 1).

4.2 MouldsType

This *ObjectType* is a container for the mould(s). It is formally defined in Table 6.

Table 6 – MouldsType Definition

Attribute	Value				
BrowseName	MouldsType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType* defined in OPC UA Part 5					
HasProperty	Variable	0:NodeVersion	String	PropertyType	M, R
HasComponent	Object	Mould_<Nr>		MouldType	OP

When instances for moulds are created, the *BrowseNames* shall be "Mould_<Nr>" (starting with 1).

4.3 PowerUnitsType

This *ObjectType* is a container for the power unit(s). It is formally defined in Table 7.

Table 7 – PowerUnitsType Definition

Attribute	Value				
BrowseName	PowerUnitsType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType* defined in OPC UA Part 5					
HasProperty	Variable	0:NodeVersion	String	PropertyType	M, R
HasComponent	Object	PowerUnit_<Nr>		PowerUnitType	OP

When instances for power units are created, the *BrowseNames* shall be “PowerUnit_<Nr>” (starting with 1).

4.4 TemperatureZones

This *ObjectType* is a container for temperature zones. It is formally defined in Table 8.

Table 8 – TemperatureZonesType Definition

Attribute	Value				
BrowseName	TemperatureZonesType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType* defined in OPC UA Part 5					
HasProperty	Variable	0:NodeVersion	String	PropertyType	M, R
HasComponent	Object	<TemperatureZone_Nr>		TemperatureZoneType	OP

When instances for temperature zones are created, the *BrowseNames* shall be "TemperatureZone_<Nr>" (starting with 1) or as specified for the subtypes of *TemperatureZonesType* (e.g. "BarrelTemperatureZone_<Nr>" for instances of *BarrelTemperatureZoneType*, see 14.10).

5 MachineInformationType

5.1 MachineInformationType Definition

This *ObjectType* represents the general description of a machine. The information is fixed by the manufacturer and not changeable by the user. It is formally defined in Table 9.

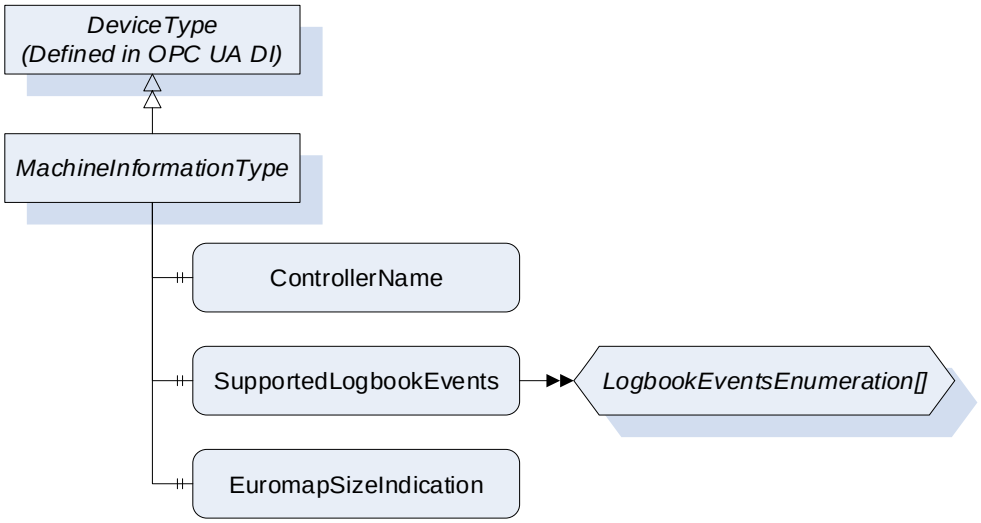


Figure 1 – MachineInformationType Overview

Table 9 – MachineInformationType Definition

Attribute	Value				
BrowseName	MachineInformationType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *DeviceType* defined in OPC UA for Devices (DI)					
HasProperty	Variable	DeviceClass	String	PropertyType	M, R
HasProperty	Variable	ControllerName	String	PropertyType	M, R
HasProperty	Variable	SupportedLogbookEvents	LogbookEventsEnumeration[]	PropertyType	M, R
HasProperty	Variable	EuromapSizeIndication	String	PropertyType	O, R

5.2 Properties included in DeviceType

The following parameters are already included in the *DeviceType* (defined in OPC UA Part 100).

5.2.1 DeviceClass

The *DeviceClass Property* indicates in which domain or for what purpose a certain *Device* is used. The *Property* is optional in OPC UA DI. Here it is overridden and made mandatory. The value is specified in the specific Companion Specification (e.g. "Injection Moulding Machine" for EUROMAP 77).

5.2.2 Manufacturer

The *Manufacturer Property* provides the name of the manufacturer of the machine (e.g. "Negri Bossi").

5.2.3 DeviceManual

The *DeviceManual Property* allows specifying an address of the user manual for the *Device*. It may be a pathname in the file system or a URL (Web address). If the manual is not directly accessible, it may also be a general web address (e.g. “netstal.com”).

5.2.4 Model

The *Model Property* represents the name of the machine type (e.g. "KM 1000-2500", "Allrounder").

5.2.5 SoftwareRevision

The *SoftwareRevision Property* represents the software version used in the control unit (e.g. "nb2001v11B030").

5.2.6 SerialNumber

The *SerialNumber Property* represents the serial number of the machine (unique ID given by the manufacturer, e.g. "1240114").

5.2.7 DeviceRevision

The *DeviceRevision Property* provides the overall revision level of the *Device*.

5.2.8 HardwareRevision

The *HardwareRevision Property* provides the revision level of the hardware of the *Device*.

5.2.9 RevisionCounter

The *RevisionCounter Property* is an incremental counter indicating the number of times the static data within the *Device* has been modified.

5.3 Additional properties

The following parameters are defined to extend the *DeviceType*.

5.3.1 ControllerName

The *ControllerName Property* represents the name of the machine controller (e.g. "MC5").

5.3.2 SupportedLogbookEvents

This list of *LogbookEventsEnumeration* gives information which *LogbookEvents* (see 6) are supported by the machine.

The *LogbookEventsEnumeration* is defined in Table 10:

Table 10 – LogbookEventsEnumeration Values

Value	Description
PARAMETER_CHANGE_0	Support of *ParameterChangeLogType* (see 6.5). The machine will fire events when production parameters are changed.
USER_1	Support of *UserLogType* (see 6.6). The machine will fire events when users log in or off at the machine control.
REMOTE_ACCESS_2	Support of *RemoteAccessLogType* (see 6.7). The machine will fire events when remote users log in or off.
SEQUENCE_CHANGE_3	Support of *SequenceChangeLogType* (see 6.8). The machine will fire events when there are changes in the production sequence.
MACHINE_MODE_CHANGE_4	Support of *MachineModeChangeLogType* (see 6.9). The machine will fire events when the machine mode is changed.
PRODUCTION_STATUS_CHANGE_5	Support of *ProductionStatusChangeLogType* (see 6.10). The machine will fire events when the production status is changed.
PRODUCTION_DATASET_CHANGE_6	Support of *ProductionDatasetChangeLogType* (see 6.11). The machine will fire events when the production dataset is changed.
PRODUCTION_DATASET_FROZEN_7	Support of *ProductionDatasetFrozenLogType* (see 6.12). The machine will fire events when the frozen status of the production dataset (see 17.3.4) is changed.
STANDSTILL_REASON_8	Support of *StandstillReasonLogType* (see 6.13). The machine will fire events when the standstill reason is changed.
MESSAGE_9	Support of *MessageLogType* (see 6.14). The machine will fire events for each fired MessageCondition.
USER_FEEDBACK_10	Support of *UserFeedbackLogType* (see 6.15). The machine will fire events for each message entered by the user.

5.3.3 EuromapSizeIndication

The *EuromapSizeIndication Property* represents the size indication of the machine in accordance with a EUROMAP recommendation (e.g. for injection moulding machines (EUROMAP 1): "3430 V – 3750").

6 LogbookEvent

6.1 LogbookEvent Definition

Logbook events are fired by the machine for the documentation of relevant changes in the machine configuration/status. The *LogbookEventType* is formally defined in Table 11.

Table 11 – LogbookEventType Definition

Attribute	Value				
BrowseName	LogbookEventType				
IsAbstract	True				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseEventType* defined in OPC UA Part 5					
HasComponent	Object	User		UserType	M
HasProperty	Variable	EventOriginator	EventOriginator Enumeration	PropertyType	M
HasProperty	Variable	JobCycleCounter	UInt64	PropertyType	O
HasSubtype	ObjectType	ParameterChangeLogType	Defined in 6.5		
HasSubtype	ObjectType	UserLogType	Defined in 6.6		
HasSubtype	ObjectType	RemoteAccessLogType	Defined in 6.7		
HasSubtype	ObjectType	SequenceChangeLogType	Defined in 6.8		
HasSubtype	ObjectType	MachineModeChangeLogType	Defined in 6.9		
HasSubtype	ObjectType	ProductionStatusChangeLogType	Defined in 6.10		
HasSubtype	ObjectType	ProductionDatasetChangeLogType	Defined in 6.11		
HasSubtype	ObjectType	ProductionDatasetFrozenLogType	Defined in 6.12		
HasSubtype	ObjectType	StandstillReasonLogType	Defined in 6.13		
HasSubtype	ObjectType	MessageLogType	Defined in 6.14		
HasSubtype	ObjectType	UserFeedbackLogType	Defined in 6.15		

The *LogbookEventType* is abstract. There will be no instances of a *LogbookEventType* itself, but there will be instances of its subtypes which provide detailed information.

The *EventSource* is the root node of the interface (e.g. instance of IMM_MES_InterfaceType for EUROMAP 77).

NOTE: The *EventSource*, *Time* and a *Message* are already included in the *BaseEventType* of OPC UA.

There shall be a buffer to be accessible via the OPC UA service history read (even when the machine has been switched off in the meantime). The minimum size is 100 *LogbookEvents* to be stored in a (persistent) ring buffer containing the recent 100 events (over all *LogbookEvents*).

6.2 User

This *Property* indicates the user who is responsible for the change that leads to the event. The fields of *UserType* shall be null or emtpy if no user is directly responsible (e.g. for messages coming from the machine control system).

6.3 EventOriginator

This *Property* represents the originator of a logbook event. The *EventOriginatorEnumeration* is defined in Table 12.

Table 12 – EventOriginatorEnumeration Definition

Value	Description
OTHER_0	Undefined
MACHINE_1	The machine causes the event (e.g. an alarm)
OPERATOR_2	The operator of the machine causes the event (e.g. a parameter change)
MES_3	The MES causes the event (e.g. a MESMessage)
PERIPHERAL_DEVICE_4	A peripheral device causes the event (e.g. an alarm)

6.4 JobCycleCounter

This *Property* represents the current value of *JobCycleCounter* in the *ActiveJobValues Object* when the event is fired (see 15.4.7.1). Only to be used for cyclic production (e.g. injection moulding).

6.5 ParameterChangeLogType

The *ParameterChangeLogType* is used for the logging of relevant changes in production parameters. The decision which parameter is relevant for the production is done by the machine.

Table 13 – ParameterChangeLogType Definition

Attribute	**Value**				
BrowseName	ParameterChangeLogType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *LogbookEventType*					
HasProperty	Variable	ParameterId	String	PropertyType	M
HasProperty	Variable	OldValue	BaseDataType	PropertyType	M
HasProperty	Variable	OldValueUnit	EUInformation	PropertyType	O
HasProperty	Variable	NewValue	BaseDataType	PropertyType	M
HasProperty	Variable	NewValueUnit	EUInformation	PropertyType	O

The *ParameterId Property* represents the Id of the changed parameter.

The *OldValue Property* represents the old value of the changed parameter.

The *NewValue Property* represents the new value of the changed parameter.

Depending on the changed parameter, the *Datatype* of *OldValue* and *NewValue* are subtypes of *BaseDataType* (String, Number, …). Where the unit is important (e.g. temperatures, lengths…), also *OldValueUnit* and *NewValueUnit* shall be used (see OPC UA Part 5 for the definition of *EUInformation*).

6.6 UserLogType

The *UserLogType* is used for logging which users are logged in to the machine.

Table 14 – UserLogType Definition

Attribute	**Value**				
BrowseName	UserLogType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *LogbookEventType*					
HasProperty	Variable	UserChange	UserChangeEnumeration	PropertyType	M

The *UserChangeEnumeration* is defined in Table 15.

Table 15 – UserChangeEnumeration Definition

Value	**Description**
LOG_ON_0	The User has logged on the machine.
LOG_OFF_1	The User has logged off the machine.

6.7 RemoteAccessLogType

The *RemoteAccessLogType* is used for logging access from outside to the machine (e.g. remote service).

Table 16 – RemoteAccessLogType Definition

Attribute	Value				
BrowseName	RemoteAccessLogType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *LogbookEventType*					
HasProperty	Variable	RemoteUserName	String	PropertyType	M
HasProperty	Variable	UserChange	UserChangeEnumeration	PropertyType	M
HasProperty	Variable	Origin	String	PropertyType	O

RemoteUserName: Name of the remote user (e.g. name of the service employee doing remote service)

UserChange: The *UserChangeEnumeration* is the same used in *UserLogType* and is defined in Table 15.

Origin: Information about the origin of the remote access (e.g. "Headquarters").

6.8 SequenceChangeLogType

The *SequenceChangeLogType* is used for the logging changes in the production sequence.

Table 17 – SequenceChangeLogType Definition

Attribute	Value				
BrowseName	SequenceChangeLogType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *LogbookEventType*					
HasProperty	Variable	SequenceChange	SequenceChangeEnumeration	PropertyType	M

The *SequenceChange Property* allows classifying the changes. (A description of the changes is included in the *Message Property* of the *LogBookEventType*)

Table 18 – SequenceChangeEnumeration Definition

Value	Description
UPDATE_0	The sequence has been updated (e.g. when a new production dataset has been activated)
ADD_1	An element has been added to the sequence
MODIFY_2	An element of the sequence has been modified.
MOVE_3	An element of the sequence has been moved.
DELETE_4	An element of the sequence has been deleted.

6.9 MachineModeChangeLogType

The *MachineModeChangeLogType* is used for logging changes of the machine mode.

Table 19 – MachineModeChangeLogType Definition

Attribute	Value				
BrowseName	MachineModeChangeLogType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *LogbookEventType*					
HasProperty	Variable	OldMachineMode	MachineModeEnumeration	PropertyType	M
HasProperty	Variable	NewMachineMode	MachineModeEnumeration	PropertyType	M

The *MachineModeEnumeration* is defined in 9.4.

6.10 ProductionStatusChangeLogType

The *ProductionStatusChangeLogType* is used for logging changes of the production status.

Table 20 – ProductionStatusChangeLogType Definition

Attribute	Value				
BrowseName	ProductionStatusChangeLogType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *LogbookEventType*					
HasProperty	Variable	OldProductionStatus	ProductionStatusEnumeration	PropertyType	M
HasProperty	Variable	NewProductionStatus	ProductionStatusEnumeration	PropertyType	M

The *ProductionStatusEnumeration* is defined in 11.7.1.

6.11 ProductionDatasetChangeLogType

The *ProductionDatasetChangeLogType* is used when a new production dataset is loaded and activated in the control system of the machine.

Table 21 – ProductionDatasetChangeLogType Definition

Attribute	Value				
BrowseName	ProductionDatasetChangeLogType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *LogbookEventType*					
HasProperty	Variable	OldProductionDatasetName	String	PropertyType	M
HasProperty	Variable	NewProductionDatasetName	String	PropertyType	M

6.12 ProductionDatasetFrozenLogType

The *ProductionDatasetFrozenLogType* is used when a production dataset is locked or unlocked (see 17.3.4).

Table 22 – ProductionDatasetFrozenLogType Definition

Attribute	Value				
BrowseName	ProductionDatasetFrozenLogType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *LogbookEventType*					
HasProperty	Variable	OldValue	Boolean	PropertyType	M
HasProperty	Variable	NewValue	Boolean	PropertyType	M

6.13 StandstillReasonLogType

The *StandstillReasonLogType* is used for logging *StandstillReasons*.

Table 23 – StandstillReasonLogType Definition

Attribute	Value				
BrowseName	StandstillReasonLogType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *LogbookEventType*					
HasProperty	Variable	StandstillReasonId	String	PropertyType	M

6.14 MessageLogType

The *MessageLogType* is used for logging *MessageConditions* (see 11.8).

Table 24 – MessageLogType Definition

Attribute	Value				
BrowseName	MessageLogType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *LogbookEventType*					
HasProperty	Variable	Id	String	PropertyType	M
HasProperty	Variable	IsStandstillMessage	Boolean	PropertyType	M
HasProperty	Variable	Classification	Enumeration	PropertyType	M

Classification: Classification of the message. The valid values of the Enumeration are specified in the specific Companion Specification (e.g. *IMMMessageClassificationEnumeration* in EUROMAP 77).

6.15 UserFeedbackLogType

The *UserFeedbackLogType* is used for logging text messages entered by the user into the machine control system.

Table 25 – UserFeedbackLogType Definition

Attribute	Value				
BrowseName	UserFeedbackLogType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *LogbookEventType*					
HasProperty	Variable	Classification	Enumeration	PropertyType	M

Classification: Classification of the message. The valid values of the Enumeration are specified in the specific Companion Specification (e.g. *IMMMessageClassificationEnumeration* in EUROMAP 77).

NOTE: The *Message* itself is included in the *BaseEventType*

7 MachineConfigurationType

7.1 MachineConfigurationType Definition

This OPC UA *ObjectType* represents the current configuration of a machine. It is formally defined in Table 26.

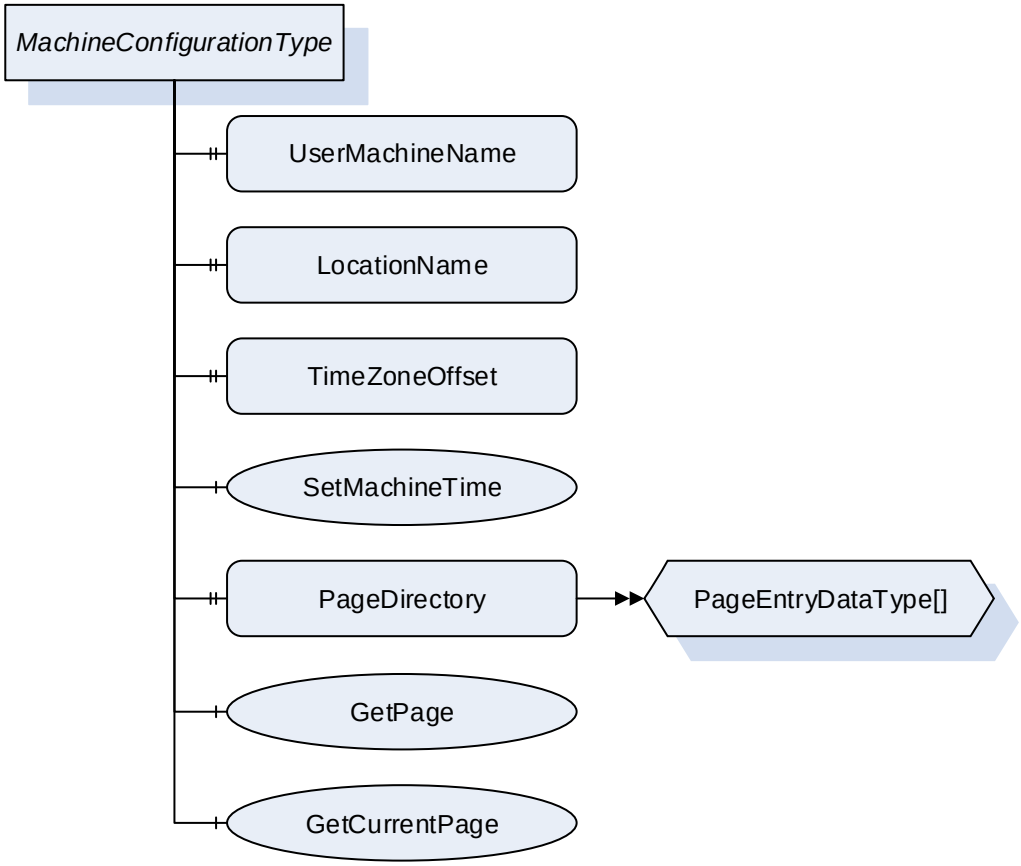


Figure 2 – *MachineConfigurationType* Overview

Table 26 – MachineConfigurationType Definition

Attribute	**Value**				
BrowseName	MachineConfigurationType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType* defined in OPC UA Part 5					
HasProperty	Variable	UserMachineName	String	PropertyType	M, RW
HasProperty	Variable	LocationName	String	PropertyType	M, RW
HasProperty	Variable	TimeZoneOffset	TimeZoneDataType	PropertyType	M, RW
HasComponent	Method	SetMachineTime			M
HasProperty	Variable	PageDirectory	PageEntryDataType[]	PropertyType	O, R
HasComponent	Method	GetPage			O
HasComponent	Method	GetCurrentPage			O

7.2 UserMachineName

The *UserMachineName Property* represents the description of the machine given by the machine operator or OPC client (e.g. "machine 42").

7.3 LocationName

The *LocationName Property* represents the description of the location of the machine given by the machine operator or OPC client (e.g. "plant 2, hall C").

7.4 TimeZoneOffset

The *TimeZoneOffset Property* represents the difference of the local time to Coordinated Universal Time (UTC) given by the machine operator or OPC client.

Information: *TimeZoneDataType* (as defined in OPC UA Part 3) is a structure with two components:

- *offset* (UInt16): Time difference from UTC in minutes (e.g. 120 for daylight saving time in Berlin)
- *daylightSavingInOffset* (Boolean): If TRUE, then daylight saving time (DST) is in effect and *offset* includes the DST correction. If FALSE, then the *offset* does not include DST correction and DST may or may not have been in effect.

NOTE: The UTC time itself is part of OPC UA Part 5: *ServerStatus → CurrentTime.*

7.5 SetMachineTime

The *SetMachineTime Method* allows setting the server time together with *TimeZoneOffset*.

Signature

```
SetMachineTime (
   [in]  DateTime              DateTime
   [in]  TimeZoneDataType      TimeZoneOffset);
```

Table 27 – SetMachineTime Method Arguments

Argument	Description
DateTime	Date and time in UTC time
TimeZoneOffset	Time difference from UTC in minutes incl. daylight saving time

Table 28 – SetMachineTime Method AddressSpace Definition

Attribute	Value				
BrowseName	SetMachineTime				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory

7.6 PageDirectory

The *PageDirectory Property* is an array and represents a list of the pages that are implemented in the machine control system and are shown on the screen of the machine. The used *PageEntryDataType* is defined in Table 29.

Table 29 – PageEntryDataType Definition

Name	Type	Description
PageEntryDataType	structure	
Id	String	Unique identifier defined by manufacturer
Title	LocalizedText	Page Name

7.7 GetPage

Method for retrieving the image of a page of the control system.

Signature

```
GetPage (
   [in]  String     Id
   [out] Image      Page);
```

Table 30 – GetPage Method Arguments

Argument	Description
Id	Id of the Page
Page	Image of a page of the control system (ImageBMP, ImageGIF, ImageJPG or ImagePNG)

Table 31 – GetPage Method AddressSpace Definition

Attribute	Value				
BrowseName	GetPage				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Mandatory

7.8 GetCurrentPage

Method for retrieving a screenshot of the control system with the currently shown contents.

Signature

```
GetCurrentPage (
   [in]  UInt32     VisualisationUnit
   [out] Image      Page);
```

Table 32 – GetCurrentPage Method Arguments

Argument	Description
VisualisationUnit	Number of the visualisation unit from which the image should be created, 0 = default unit
Page	Image of a page of the control system (ImageBMP, ImageGIF, ImageJPG or ImagePNG)

Table 33 – GetCurrentPage Method AddressSpace Definition

Attribute	Value				
BrowseName	GetCurrentPage				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Mandatory

8 MachineMESConfigurationType

8.1 MachineMESConfigurationType Definition

This OPC UA *ObjectType* represents the current configuration of a machine related to a Manufacturing Execution System (MES). It is formally defined in Table 34.

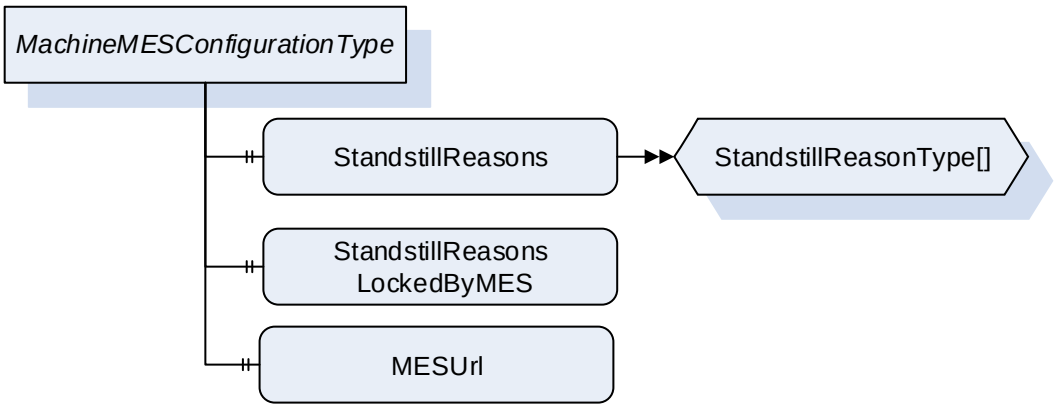


Figure 3 – *MachineMESConfigurationType* Overview

Table 34 – MachineMESConfigurationType Definition

Attribute	Value				
BrowseName	MachineMESConfigurationType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType* defined in OPC UA Part 5					
HasProperty	Variable	StandstillReasons	StandstillReasonType[]	PropertyType	M, RW
HasProperty	Variable	StandstillReasonsLockedByMES	Boolean	PropertyType	M, RW
HasProperty	Variable	MESUrl	String	PropertyType	O, RW

8.2 StandstillReasons

The *StandstillReasons Property* represents an array with a list of the standstill reasons from which one is selected by the operator in the case of a standstill. The used *StandstillReasonType* is defined in Table 35.

Table 35 – StandstillReasonType Definition

Name	Type	Description
StandstillReasonType	structure	
Id	String	Id of the standstill reason
Text	LocalizedText	Text of the standstill reason
LockedByMES	Boolean	*LockedByMES* means that this *StandstillReason* has been set or modified by the MES and so this may not be changed by the machine.

StandstillReasons shall support at least 10 entries.

The MES shall be prepared, that the list includes standstill reasons not defined by the MES (e.g. added by the operator on the machine). The user of the interface shall ensure unique Ids among all standstill reasons. To ensure consistent statistics, a change of the meaning of already assigned standstill reasons should be avoided.

8.3 StandstillReasonsLockedByMES

The *StandstillReasonsLockedByMES Property* indicates if the list *StandstillReasons* has been modified by the MES and may not be changed by the machine.

NOTE: The parameter *LockedByMES* in an item only locks this *StandstillReason* whereas *StandstillReasonsLockedByMES* locks the complete array (that means also that no items can be added by the machine).

8.4 MESUrl

The *MESUrl Property* represents a URL to display a webpage generated by the MES in a web browser integrated in the machine.

9 MachineStatusType

9.1 MachineStatusType Definition

This *ObjectType* represents the current status of a machine. It is formally defined in Table 36.

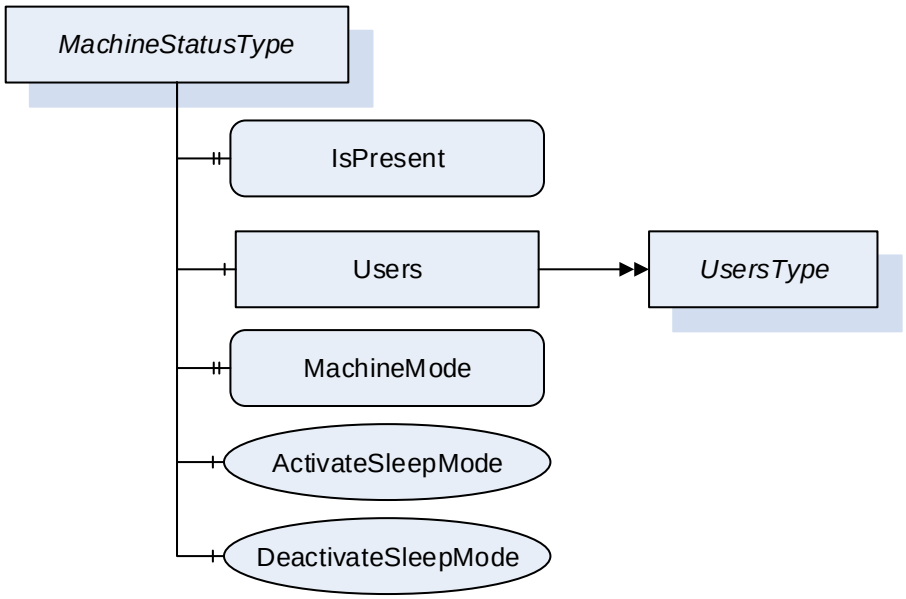


Figure 4 – *MachineStatusType* Overview

Table 36 – MachineStatusType Definition

Attribute	**Value**				
BrowseName	MachineStatusType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType* defined in OPC UA Part 5					
HasProperty	Variable	IsPresent	Boolean	PropertyType	M, R
HasComponent	Object	Users		UsersType	M
HasProperty	Variable	MachineMode	MachineModeEnumeration	PropertyType	M, R
HasComponent	Method	ActivateSleepMode			O
HasComponent	Method	DeactivateSleepMode			O

9.2 IsPresent

This *Property* informs the client if the machine is physically present and connected. May be FALSE e.g. when an aggregation server is configured to provide data for several machines.

9.3 Users

This *Object* is a container for the user(s) of the machine (see 4.1). The *UserType* itself is defined in chapter 10.

9.4 MachineMode

The *MachineMode Property* represents the current machine mode (as defined by mode selector on the machine). The *MachineModeEnumeration* is defined in Table 37:

Table 37 – MachineModeEnumeration Values

Value	Description
OTHER_0	This state is used if none of the other states below apply.
AUTOMATIC_1	The machine is in automatic mode.
SEMI_AUTOMATIC_2	The machine is in semi-automatic mode.
MANUAL_3	The machine is in manual mode.
SETUP_4	The machine is in setup mode.
SLEEP_5	The machine is in sleep mode. Machine is still switched on, energy consumption reduced by e.g. reducing heating, switching drives off. Production is not possible.

9.5 ActivateSleepMode, DeactivateSleepMode

These two *Methods* allow the client (e.g. MES) to activate or deactivate the sleep mode of the machine if provided.

Activation of sleep mode sets the *MachineMode* (see 9.4) to SLEEP_5.

Signatures

```
ActivateSleepMode ();
DeactivateSleepMode ();
```

The methods have no *Input*- or *OutputArguments*.

Table 38 – ActivateSleepMode Method AddressSpace Definition

Attribute	Value				
BrowseName	ActivateSleepMode				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**

Table 39 – DeactivateSleepMode Method AddressSpace Definition

Attribute	Value				
BrowseName	DeactivateSleepMode				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**

10 UserType

The *UserType* represents information on the operator(s) of the machine. It is formally defined in Table 40.

Table 40 –UserType Definition

Attribute	**Value**				
BrowseName	UserType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType* defined in OPC UA Part 5					
HasProperty	Variable	Id	String	PropertyType	M, R
HasProperty	Variable	Name	String	PropertyType	M, R
HasProperty	Variable	IsPresent	Boolean	PropertyType	M, R
HasProperty	Variable	CardUid	String	PropertyType	O, R
HasProperty	Variable	UserLevel	String	PropertyType	O, R
HasProperty	Variable	UserRole	String	PropertyType	O, R
HasProperty	Variable	Language	LocaleId	PropertyType	O, R

All fields may contain empty strings if not supported.

NOTE: The information provided via the UserType (especially Id and Name) might affect data protection laws and agreements with the employees might be necessary before usage. In case of doubt anonymized information should be provided.

10.1 Id

The *Id Property* represent the Id of the user.

10.2 Name

The *Name Property* represent the Name of the user.

10.3 IsPresent

The machine can have instances for the maximum number of users that **can** be simultaneously logged in. TRUE if the instance of *UserType* represents a user that is currently logged in.

10.4 CardUid

This *Property* represents the Uid of the identification card used by the operator for logging in to the machine.

NOTE: The variables *Name*, *Id* and *CardUid* are in accordance with the user identification as described in EUROMAP 65.

10.5 UserLevel

The *UserLevel Property* represent the level of the user (e.g. "1", "2"). The possible values are defined by the manufacturer of the machine.

NOTE: In this *Property*, the Acess rights as described in EUROMAP 65 can be stored.

10.6 UserRole

The *UserRole Property* represents the role of the user (e.g. "Administrator"). The possible values are defined by the manufacturer of the machine.

10.7 Language

The *Language Property* represents the currently selected language on the machine control unit. Indication of language with Language code = Alpha-3 (three-letter) code according to ISO 639-2/B and Country code = Alpha-2 (two-letter) code according to ISO 3166-1 (e.g. "eng-US"; see EUROMAP 65)

11 MachineMESStatusType

11.1 MachineMESStatusType Definition

This *ObjectType* represents the current status of a machine related to the MES. It is formally defined in Table 41.

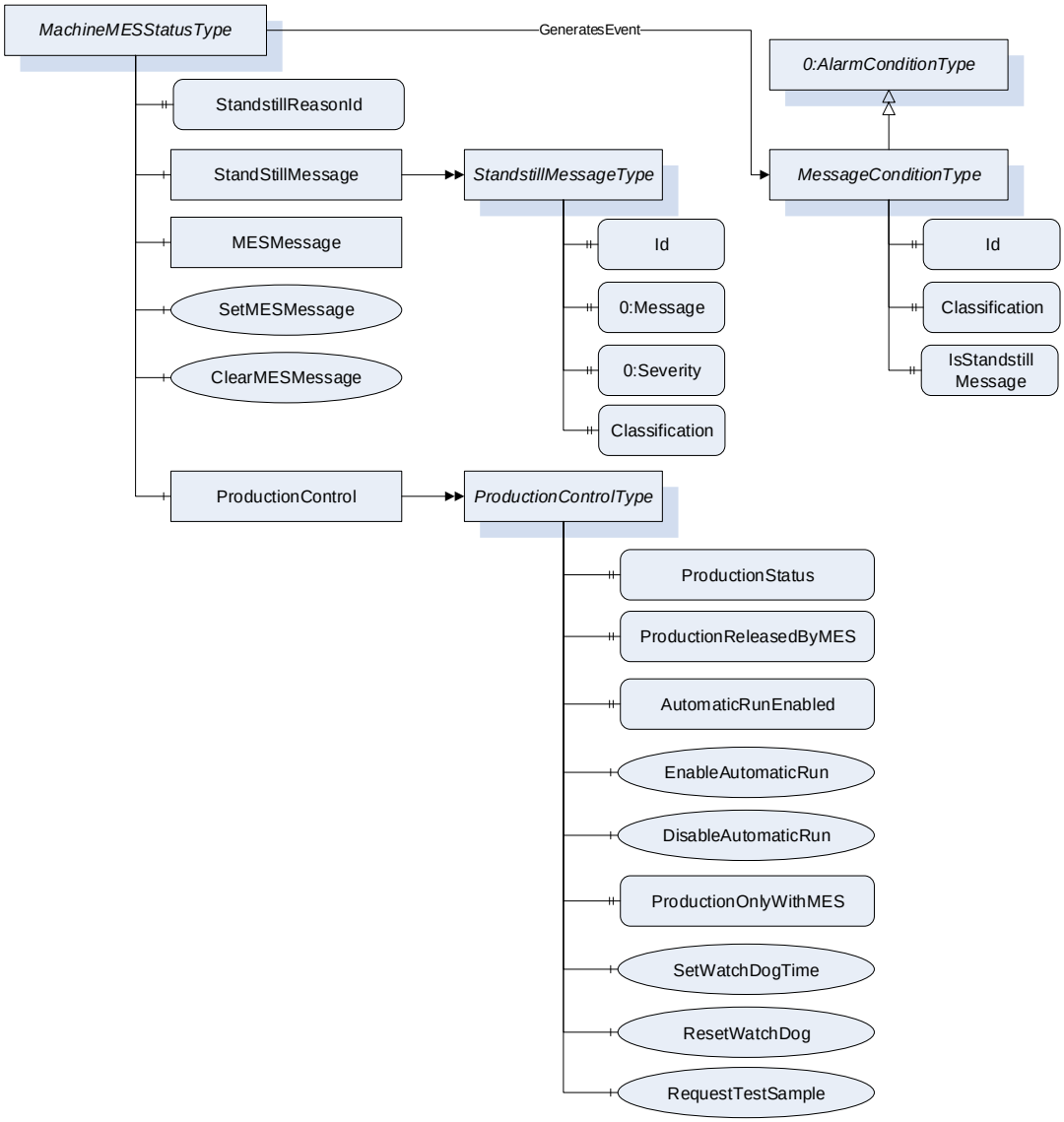


Figure 5 – *MachineMESStatusType* Overview

Table 41 – MachineMESStatusType Definition

Attribute	Value				
BrowseName	MachineMESStatusType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType* defined in OPC UA Part 5					
HasProperty	Variable	StandstillReasonId	String	PropertyType	M, R
HasComponent	Object	StandstillMessage		StandStillMessageType	M
HasComponent	Object	MESMessage		MESMessageType	M
HasComponent	Method	SetMESMessage			M
HasComponent	Method	ClearMESMessage			M
HasComponent	Object	ProductionControl		ProductionControlType	M
GeneratesEvent	ObjectType	MessageConditionType	Defined in 11.8		

11.2 StandstillReasonId

The *StandstillReasonId Property* represents the Id of the *StandstillReason* (see *MachineConfiguration*) set by the operator after a standstill occurs.

Default value: empty string (= no active standstill reason).

Set to an empty string by machine with starting of production.

11.3 StandstillMessage

The *StandstillMessage Object* represents the fault which causes standstill. This is set by machine control.

The *StandstillMessageType* is formally defined in Table 42.

Table 42 – StandStillMessageType Definition

Attribute	Value				
BrowseName	StandStillMessageType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType* defined in OPC UA Part 5					
HasProperty	Variable	Id	String	PropertyType	M, R
HasProperty	Variable	0:Message	LocalizedText	PropertyType	M, R
HasProperty	Variable	0:Severity	UInt16	PropertyType	M, R
HasProperty	Variable	Classification	Enumeration	PropertyType	M, R

Classification: Classification of the message. The valid values of the Enumeration are specified in the specific Companion Specification (e.g. IMMMessageClassificationEnumeration in EUROMAP 77).

NOTE: When creating an instance of the *StandstillMessage* object, the *DataType* of *Classification* shall be set to the specificc *Enumeration* subtype (e.g. *IMMMessageClassificationEnumeration*).

11.4 MESMessage

The *MESMessage Object* represents a text message sent from the MES to be shown on the machine. The *Properties* of this *Object* shall only be set/changed by the MES OPC UA Client.

NOTE: This Property shall not be writeable directly. The method *SetMESMessage* is used for the modification.

Table 43 – MESMessageType Definition

Attribute	Value				
BrowseName	MESMessageType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType* defined in OPC UA Part 5					
HasProperty	Variable	Id	String	PropertyType	M, R
HasProperty	Variable	Message	String	PropertyType	M, R
HasProperty	Variable	0:Severity	UInt16	PropertyType	M, R

11.5 SetMESMessage

Method for setting the *MESMessage*.

Signature

```
SetMESMessage (
   [in]  String     Id
   [in]  String     Message
   [in]  UInt16     Severity);
```

Table 44 – SetMESMessage Method Arguments

Argument	Description
Id	Id of the Message (can e.g. be used for automatic processing of the message)
Message	Text of Message
Severity	Severity as defined in the *BaseEventType* (1 = low – 1000 = high)

Table 45 – SetMESMessage Method AddressSpace Definition

Attribute	Value				
BrowseName	SetMESMessage				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory

11.6 ClearMESMessage

Method for clearing the *MESMessage*. Calling this method sets the Properties *Id* and *Message* in *MESMessage* to empty string and *Severity* to 0.

Signature

```
ClearMESMessage ();
```

The method has no *Input-* or *OutputArguments*.

Table 46 – ClearMESMessage Method AddressSpace Definition

Attribute	Value				
BrowseName	ClearMESMessage				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**

11.7 ProductionControlType

The *ProductionControl Object* allows the MES to control the production of the machine. It is formally defined in Table 47.

Table 47 – ProductionControlType Definition

Attribute	Value				
BrowseName	ProductionControlType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType* defined in OPC UA Part 5					
HasProperty	Variable	ProductionStatus	ProductionStatusEnumeration	PropertyType	M, R
HasProperty	Variable	ProductionReleasedByMES	Boolean	PropertyType	M, RW
HasProperty	Variable	AutomaticRunEnabled	Boolean	PropertyType	M, R
HasComponent	Method	EnableAutomaticRun			M
HasComponent	Method	DisableAutomaticRun			M
HasProperty	Variable	ProductionOnlyWithMES	Boolean	PropertyType	O, RW
HasComponent	Method	SetWatchDogTime			O
HasComponent	Method	ResetWatchDog			O
HasComponent	Method	RequestTestSample			O

11.7.1 ProductionStatus

The *ProductionStatus Property* of *DataType ProductionStatusEnumeration* represents the production status when the machine is in automatic or semi-automatic mode. When the machine is in another mode, the value is not relevant (no production). The *ProductionStatusEnumeration* is formally defined in Table 48.

Table 48 – ProductionStatusEnumeration Definition

Value	Description
OTHER_0	This state is used if none of the other states below apply.
NO_PRODUCTION_1	The machine does not produce any parts/products.
START_UP_2	The machine is producing parts/products in the start-up phase. So the correct settings of the machines are not reached.
READY_FOR_PRODUCTION_3	The machine is producing parts/products, the correct settings of the machines are reached but the production is not yet released (e.g. waiting for release from quality assurance).
PRODUCTION_4	The machine is producing parts/products. In semi-automatic mode also during waiting time (e.g. for manual loading/unloading of parts) ProductionStatus remains in this state (time out possible if e.g. cycle time exceeds a pre-defined limit).
DRY_RUN_5	The machine is moving without material.

NOTE: This is a list of a possible status and not a sequence. The support of the values NO_PRODUCTION_1 and PRODUCTION_4 is mandatory. The use of the other values is optional.

The *ProductionStatus* is set by the machine/operator. The selection of the value PRODUCTION_4 can be prevented by the MES OPC UA client by setting *ProductionReleasedByMES* to FALSE.

11.7.2 ProductionReleasedByMES

The *ProductionReleasedByMES Property* indicates if *ProductionStatus* may have the value PRODUCTION_4. Default value is TRUE. If *ProductionReleased* is FALSE it also prevents that *JobStatus* (see 15.4.1) has the value JOB_IN_PRODUCTION_6.

If *ProductionReleased* is set from TRUE to FALSE when *ProductionStatus* has the value PRODUCTION_4, the value shall change to READY_FOR_PRODUCTION_3. If *JobStatus* has the value JOB_IN_PRODUCTION_6 it shall change to JOB_INTERRUPTED_7.

11.7.3 AutomaticRunEnabled, EnableAutomaticRun, DisableAutomaticRun

The *AutomaticRunEnabled Property* indicates if semi-automatic and automatic run of the machine is allowed by MES. If FALSE, the machine cannot start in semi-automatic or automatic mode.
The value is set to TRUE by the *Method EnableAutomaticRun* and to FALSE by the *Method DisableAutomaticRun.* The default value is TRUE.

Can e.g. be used if wrong mould is installed or to force the operator to enter a *StandstillReason* after machine stop.

If the machine is running in (semi-)automatic, a call of *DisableAutomaticRun* will stop the machine (if applicable at the end of the cycle). The machine should show a message to the operator why the machine has stopped.

Signatures

```
EnableAutomaticRun ();
DisableAutomaticRun ();
```

The methods have no *Input*- or *OutputArguments*.

Table 49 – EnableAutomaticRun Method AddressSpace Definition

Attribute	Value				
BrowseName	EnableAutomaticRun				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**

Table 50 – DisableAutomaticRun Method AddressSpace Definition

Attribute	Value				
BrowseName	DisableAutomaticRun				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**

11.7.4 ProductionOnlyWithMES

The optional *Property ProductionOnlyWithMES* indicates if production with the machine is only allowed when the MES is active. When *ProductionOnlyWithMES* is TRUE and the connection to the MES is lost, the machine may not have the *ProductionStatus* PRODUCTION_4 and not the *JobStatus* JOB_IN_PRODUCTION_6 (see 15.4.1).

NOTE: It is not fixed by this specification if the machine stops or continues running.

The default value is FALSE.

Only one MES client shall write this Property.

11.7.5 SetWatchDogTime, ResetWatchDog

Some production jobs need 100% documentation of the production parameters. To ensure this, a *WatchDog* can be used. By setting the *WatchDog* time with the Method *SetWatchDogTime* the production is only released for the given time. The *Method ResetWatchDog* sets the timer to the value set by the last calling of *SetWatchDogTime*. This indicates to the machine that the MES is still connected and able to store the production parameters.

Only one MES client shall call these *Methods* to avoid overlapping.

When the defined time is exceeded without reset, the machine may not have the *ProductionStatus* PRODUCTION_4 and not the *JobStatus* JOB_IN_PRODUCTION_6 (see 15.4.1).

NOTE: It is not fixed by this specification if the machine stops or continues running.

Signature

```
SetWatchDogTime (
   [in] Int32 WatchDogTime);
```

Table 51 – SetWatchDogTime Method Arguments

Argument	Description
WatchDogTime	Time in seconds for which production is enabled by the watch dog

Calling the method with *WatchDogTime* = -1 disables the watch dog and the machine can stay in production.

Table 52 – SetWatchDogTime Method AddressSpace Definition

Attribute	Value				
BrowseName	SetWatchDogTime				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory

Signature

```
ResetWatchDog ();
```

The method has no *Input*- or *OutputArguments*.

Table 53 – ResetWatchDog Method AddressSpace Definition

Attribute	Value				
BrowseName	ResetWatchDog				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**

11.7.6 RequestTestSample

The machine shall separate a test sample (e.g. for quality check). The size of the test sample depends on the product/machine configuration.

Signature

```
RequestTestSample ();
```

The method has no *Input*- or *OutputArguments*.

Table 54 – RequestTestSample Method AddressSpace Definition

Attribute	Value				
BrowseName	RequestTestSample				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**

11.8 MessageConditionType

The *MessageConditions* represent text messages (incl. error messages) of the control system currently shown on the screen of the machine. The *MessageConditionType* is formally defined in Table 55.

Table 55 – MessageConditionType Definition

Attribute	**Value**				
BrowseName	MessageConditionType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *AlarmConditionType* defined in OPC UA Part 9					
HasProperty	Variable	Id	String	PropertyType	M, R
HasProperty	Variable	Classification	Enumeration	PropertyType	M, R
HasProperty	Variable	IsStandstillMessage	Boolean	PropertyType	M, R

Id: Id of the message

Classification: Classification of the message. The valid values of the Enumeration are specified in the specific Companion Specification (e.g. *IMMMessageClassificationEnumeration* in EUROMAP 77).

IsStandstillMessage: Indication if the message has led to a standstill.

NOTE: The *AlarmConditionType* is a subtype of *BaseEventType* and therefore includes the Properties *Severity* and *Message* (= text).

12 MouldType

12.1 MouldType Definition

This *ObjectType* represents the description and status of a mould (e.g. used in injection or blow moulding machines). It is formally defined in Table 56.

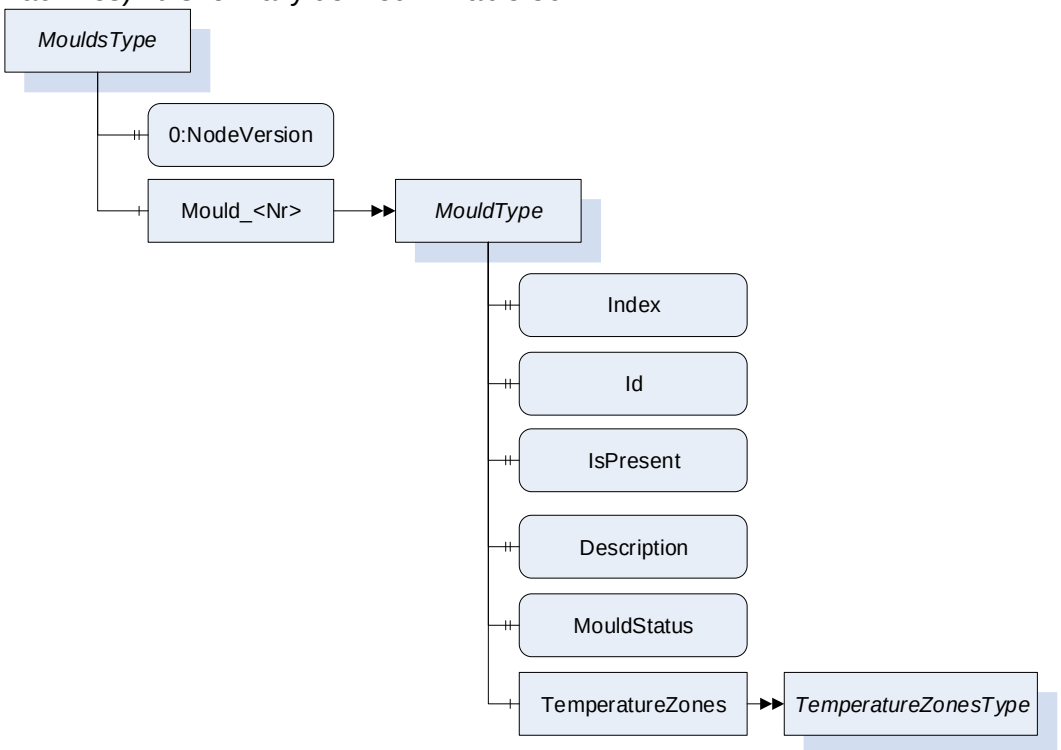


Figure 6 – *MouldType* Overview

Table 56 – MouldType Definition

Attribute	Value				
BrowseName	MouldType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType* defined in OPC UA Part 5					
HasProperty	Variable	Index	UInt32	PropertyType	M, R
HasProperty	Variable	Id	String	PropertyType	M, R
HasProperty	Variable	IsPresent	Boolean	PropertyType	M, R
HasProperty	Variable	Description	String	PropertyType	M, R
HasProperty	Variable	MouldStatus	MouldStatusEnumeration	PropertyType	M, R
HasComponent	Object	TemperatureZones		TemperatureZonesType	M

12.2 Index

The *Index Property* gives the number of the mould.

12.3 Id

The *Id Property* represents the Id of the installed mould. It is valid for a complete configuration of a mould incl. frames/inserts. If an insert is changed or deactivated, the mould gets a new *Id*.

12.4 IsPresent

This *Property* informs the client if the mould is physically present and connected. May be FALSE e.g. when instances for possible moulds are created (depending on the capabilities/connectors of the machine) which are currently not used.

12.5 Description

The *Description Property* represents a description of the installed mould.

12.6 MouldStatus

The *MouldStatusProperty* represents the current (physical) status of the mould related to the object instance. The *MouldStatusEnumeration* is defined in Table 57:

Table 57 – MouldStatusEnumeration Values

Value	Description
OTHER_0	This state is used if none of the other states below apply.
MOULD_NOT_INSTALLED_1	The mould is not installed on the machine.
MOULD_CHANGE_2	During installation or changing of the mould.
MOULD_INSTALLED_3	The mould is installed and ready for production.

12.7 TemperatureZones

This *Object* is a container for the temperature zones of the mould zones. It is formally defined in 0. Inside the container the *MouldTemperatureZoneType* as defined in 14.10.2 shall be used.

13 PowerUnitType

13.1 PowerUnitType Definition

This *ObjectType* represents information on the hydraulic units and electric drives. It is formally defined in Table 58.

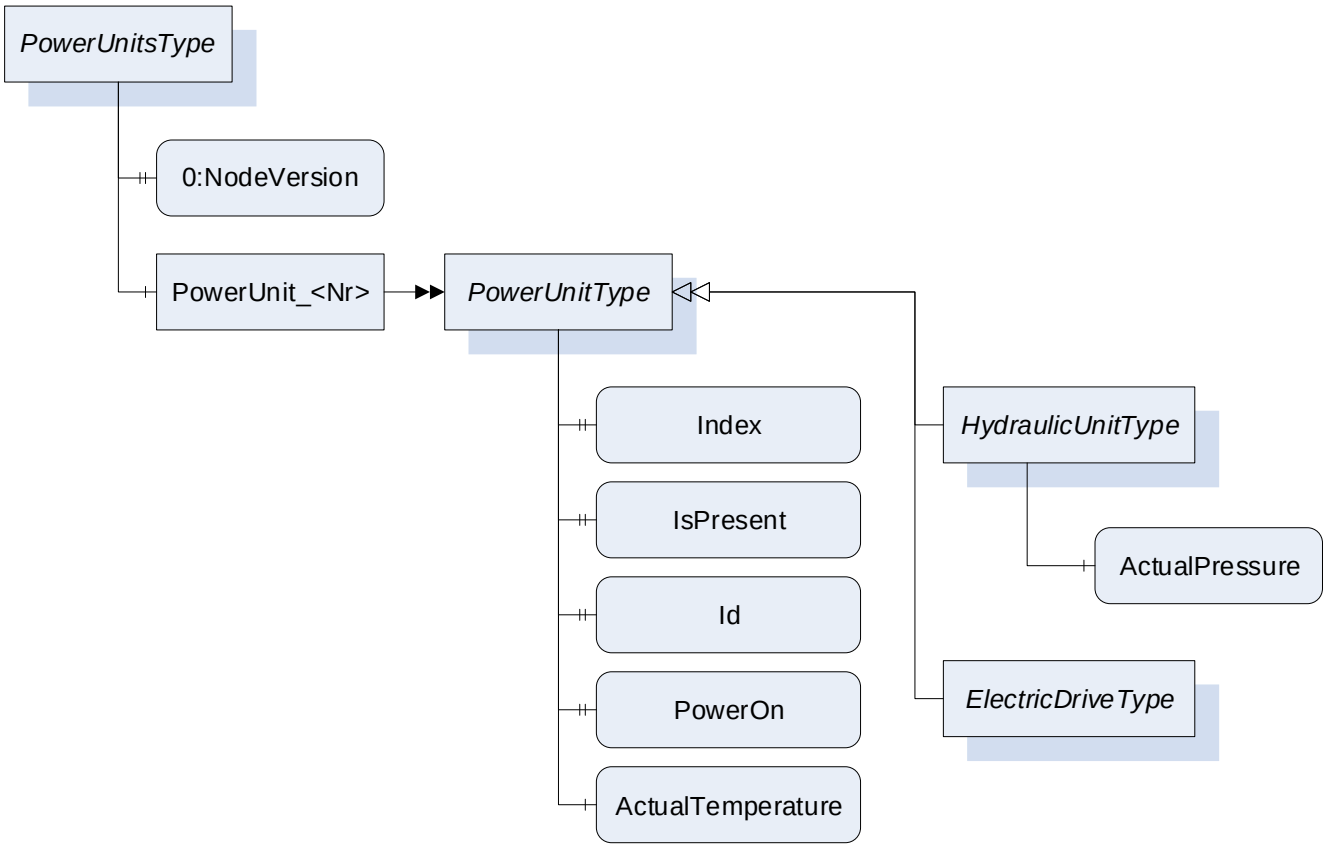


Figure 7 – *PowerUnitType* Overview

Table 58 – PowerUnitType Definition

Attribute	**Value**				
BrowseName	PowerUnitType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType* defined in OPC UA Part 5					
HasProperty	Variable	Index	UInt32	PropertyType	M, R
HasProperty	Variable	IsPresent	Boolean	PropertyType	M, R
HasProperty	Variable	Id	String	PropertyType	M, R
HasProperty	Variable	PowerOn	Boolean	PropertyType	M, R
HasComponent	Variable	ActualTemperature	Double	AnalogItemType	O, R
HasSubtype	ObjectType	HydraulicUnitType	Defined in 13.7.1		
HasSubtype	ObjectType	ElectricDriveType	Defined in 13.7.2		

13.2 Index

The *Index Property* gives the number of the power unit.

13.3 IsPresent

This *Property* informs the client if the power unit is physically present and connected. May be FALSE e.g. when instances for possible power units are created (depending on the capabilities/connectors of the machine) which are currently not used.

13.4 Id

The *Id Property* represents the Id of the *PowerUnit*.

13.5 PowerOn

The *PowerOn Property* indicates if the *PowerUnit* is switched on.

13.6 ActualTemperature

The *ActualTemperature Variable* represents the current temperature of the *PowerUnit*.

13.7 Subtypes of PowerUnitType

There are two subtypes: *HydraulicUnitType* and *ElectricDriveType*.

13.7.1 HydraulicUnitType

Table 59 – HydraulicUnitType Definition

Attribute	**Value**				
BrowseName	HydraulicUnitType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *PowerUnitType*					
HasComponent	Variable	ActualPressure	Double	AnalogItemType	O, R

The *ActualPressure Variable* represents the current pressure of the hydraulic unit.

13.7.2 ElectricDriveType

Table 60 – ElectricDriveType Definition

Attribute	**Value**				
BrowseName	ElectricDriveType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *PowerUnitType*					

This subtype has no additional child elements.

14 TemperatureZoneType

14.1 TemperatureZoneType Definition

This *ObjectType* represents a temperature zone e.g. on moulds and barrels. It is formally defined in Table 61.

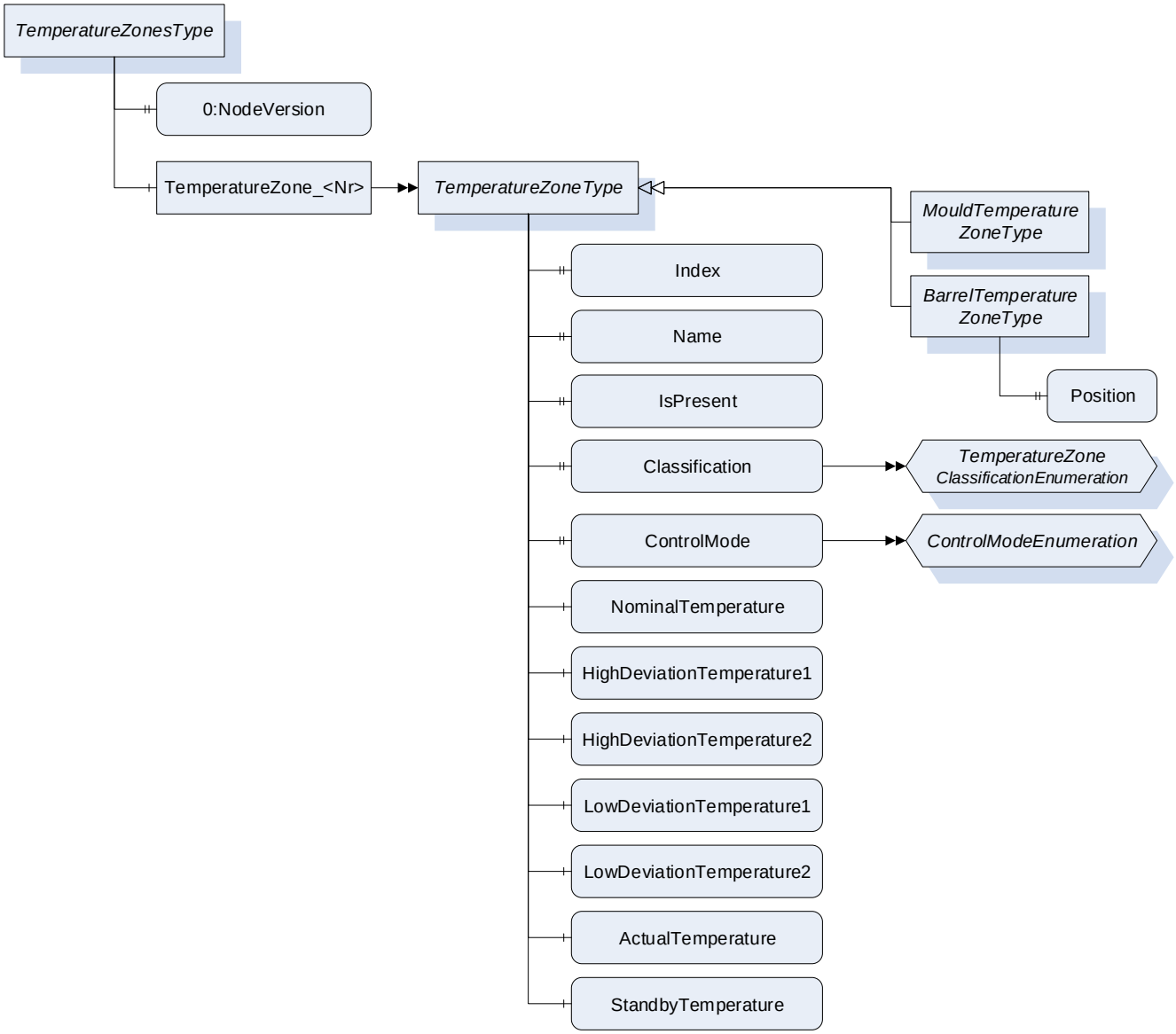


Figure 8 – *TemperatureZoneType* Overview

Table 61 – TemperatureZoneType Definition

Attribute	Value				
BrowseName	TemperatureZoneType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType* defined in OPC UA Part 5					
HasProperty	Variable	Index	UInt32	PropertyType	M, R
HasProperty	Variable	Name	String	PropertyType	M, R
HasProperty	Variable	IsPresent	Boolean	PropertyType	M, R
HasProperty	Variable	Classification	TemperatureZone ClassificationEnumeration	PropertyType	O, R
HasProperty	Variable	ControlMode	ControlModeEnumeration	PropertyType	M, R
HasComponent	Variable	NominalTemperature	Double	AnalogItemType	M, R
HasComponent	Variable	HighDeviationTemperature1	Double	AnalogItemType	O, R
HasComponent	Variable	HighDeviationTemperature2	Double	AnalogItemType	O, R
HasComponent	Variable	LowDeviationTemperature1	Double	AnalogItemType	O, R
HasComponent	Variable	LowDeviationTemperature2	Double	AnalogItemType	O, R
HasComponent	Variable	ActualTemperature	Double	AnalogItemType	M, R
HasComponent	Variable	StandbyTemperature	Double	AnalogItemType	O, R
HasSubtype	ObjectType	BarrelTemperatureZoneType	Defined in 14.10.1		
HasSubtype	ObjectType	MouldTemperatureZoneType	Defined in 14.10.2		

14.2 Index

The *Index Property* gives the number of the zone.

NOTE: If the *Property Classification* is used, for each temperature zone type the *Index* may start with 1.

14.3 Name

The *Name Property* represents the name of the zone.

14.4 IsPresent

This *Property* informs the client if the temperature zone is physically present and connected. May be FALSE e.g. when instances for possible temperature zones are created (depending on the capabilities/connectors of the machine) which are currently not used.

14.5 Classification

This *Property* informs the client about the type of the temperature zone. The *TemperatureZoneClassificationEnumeration* is defined in Table 62.

Table 62 – TemperatureZoneClassificationEnumeration Definition

Value	Description
OTHER_0	This value is used if none of the other values below apply.
HEATING_1	The zone is a heating zone
COOLING_2	The zone is a cooling zone
TEMPERATURE_CONTROL_3	The zone is controlled by a temperature control device
HOT_RUNNER_4	The zone is a hot runner zone
MEASURING_5	The zone has no heating or cooling, Only the temperature is measured.

14.6 ControlMode

The *ControlMode Property* indicates how the temperature is currently controlled. The *ControlModeEnumeration* is defined in Table 63.

Table 63 – ControlModeEnumeration Definition

Value	Description
OTHER_0	This state is used if none of the other states below apply.
OFF_1	Control is switched off.
AUTOMATIC_2	The parameter is controlled automatically.
TUNING_3	Optimisation of the control circuit.
STANDBY_4	Parameter is controlled to stand by value.
OPEN_LOOP_5	Open loop control is used.
ONLY_MEASUREMENT_6	The sensors deliver the current value but there is no controlling.

14.7 Temperatures

The following five temperatures for monitoring and controlling are defined:

Table 64 – Temperatures in TemperatureZoneType

BrowseName	Description	Optional
NominalTemperature	Nominal value (absolute) (e.g. 200°C)	No
HighDeviationTemperature1	Maximum value that is in the normal tolerance. A higher actual value may create a warning. Used for quality control. Relative value (e.g. +10°C).	Yes
LowDeviationTemperature1	Minimum value that is in the normal tolerance. A lower actual value may create a warning. Used for quality control. Relative value (e.g. -15°C).	Yes
HighDeviationTemperature2	Maximum tolerable value. A higher actual value may create an alarm. Relative value (e.g. +30°C).	Yes
LowDeviationTemperature2	Minimum tolerable value. A lower actual value may create an alarm. Relative value (e.g. -25°C).	Yes

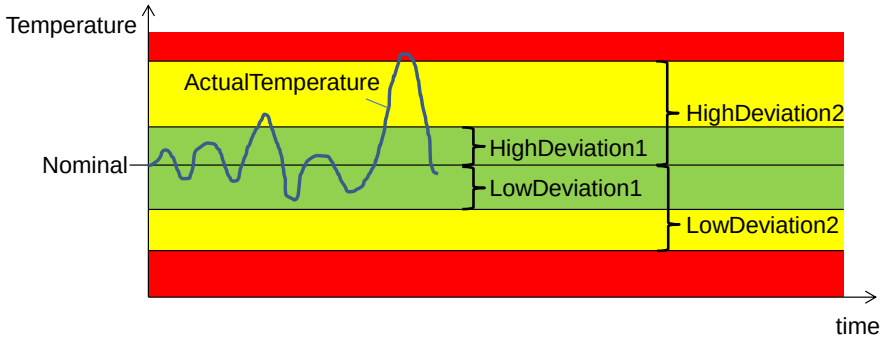


Figure 9 – Nominal and deviation temperatures in TemperatureZoneType

14.8 ActualTemperature

The *ActualTemperature Property* represents the current temperature of the *Zone*.

14.9 StandbyTemperature

The *StandbyTemperature Property* represents the standby temperature of the *Zone*.

14.10 Subtypes of TemperatureZoneType

The *TemperatureZoneType* has two subtypes to distinguish between temperature zones on barrels and in moulds.

14.10.1 BarrelTemperatureZoneType

Table 65 – BarrelTemperatureZoneType Definition

Attribute	Value				
BrowseName	BarrelTemperatureZoneType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *TemperatureZoneType*					
HasProperty	Variable	Position	UInt32	PropertyType	M, R

The *Position Property* represents the location of the temperature zone on a barrel. Counting starts with 1 beginning from the feeding. The highest position is at the nozzle.
When instances are created the *BrowseNames* shall be “BarrelTemperatureZone_<Nr>” (starting with 1).

14.10.2 MouldTemperatureZoneType

Table 66 – MouldTemperatureZoneType Definition

Attribute	Value				
BrowseName	MouldTemperatureZoneType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *TemperatureZoneType*					

The subtype has no additional child elements.
When instances are created the *BrowseNames* shall be “MouldTemperatureZone_<Nr>” (starting with 1).

NOTE: Each *MouldTemperatureZone* belongs to a *Mould*. Numbering starts with 1 for each *Mould*. That means Mould_1 has a MouldTemperatureZone_1 and Mould_2 has also a MouldTemperatureZone_1.

15 JobsType

This clause defines *ObjectsTypes* for managing production jobs on the machine and for information on their status including process parameters (temperatures, pressures…).

15.1 JobsType Definition

This *ObjectType* represents the jobs (order to produce parts/process material) stored on the machine. It is formally defined in Table 67.

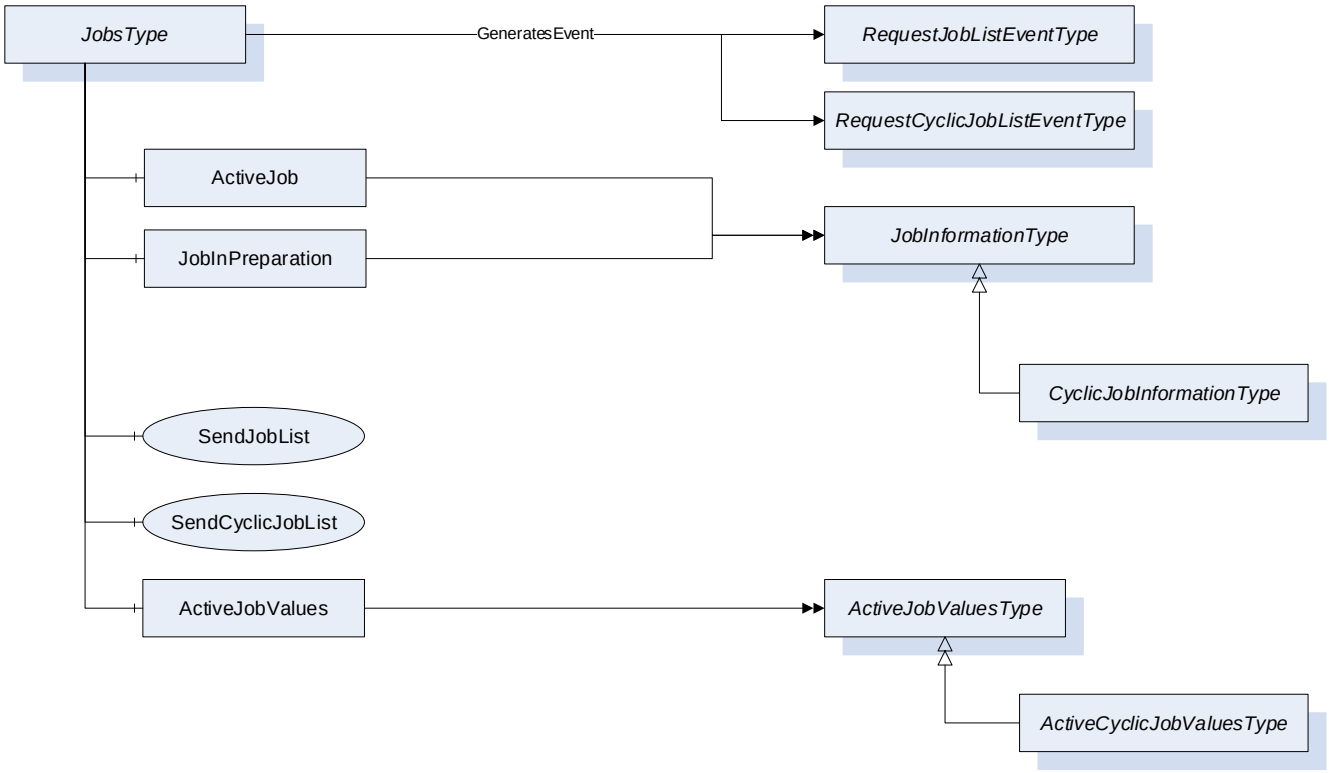


Figure 10 – JobsType Overview

Table 67 – JobsType Definition

Attribute	**Value**				
BrowseName	JobsType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType* defined in OPC UA Part 5					
HasComponent	Object	ActiveJob		JobInformationType	M
HasComponent	Object	JobInPreparation		JobInformationType	O
HasComponent	Method	SendJobList			O
HasComponent	Method	SendCyclicJobList			O
HasComponent	Object	ActiveJobValues		ActiveJobValuesType	M
GeneratesEvent	ObjectType	RequestJobListEventType	Defined in 15.3.2		
GeneratesEvent	ObjectType	RequestCyclicJobListEventType	Defined in 15.3.4		

ActiveJob represents the job that is currently active on the machine.

JobInPreparation represents a job that is in preparation. Background: Some manufacturers use two "layers" for job information in their control systems: one active information layer and one for preparation. So during a running job, inputs for the following job are possible. By pressing a button on the control panel, data from the prepared job information can be set active.

ActiveJobValues represents the status of the current job. See 15.2.11.1.

SendJobList sends a list of jobs available on the client to the server. Can be triggered by the machine by the event *RequestJobList*. See 15.4.7. For cyclic processes *SendCyclicJobList* and *RequestCyclicJobListEventType* are used.

NOTE: A method *SendContinuousJobList* for continuous processes will be added in the future.

15.2 JobInformationType

15.2.1 JobInformationType Definition

This *ObjectType* represents the data of the job. It is formally defined in Table 68.

Table 68 – JobInformationType Definition

Attribute	**Value**				
BrowseName	JobInformationType				
IsAbstract	True				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType* defined in OPC UA Part 5					
HasProperty	Variable	JobName	String	PropertyType	M, R
HasProperty	Variable	JobDescription	String	PropertyType	M, R
HasProperty	Variable	CustomerName	String	PropertyType	M, R
HasProperty	Variable	ProductionDatasetName	String	PropertyType	M, R
HasProperty	Variable	ProductionDatasetDescription	String	PropertyType	M, R
HasProperty	Variable	Material	String**[]**	PropertyType	M, R
HasProperty	Variable	ProductName	String**[]**	PropertyType	M, R
HasProperty	Variable	ProductDescription	String**[]**	PropertyType	M, R
HasProperty	Variable	ContinueAtJobEnd	Boolean	PropertyType	M, RW
HasSubtype	ObjectType	CyclicJobInformationType			

The *JobInformationType* is abstract. A derived concrete type, which includes a method for writing the *Properties*, must be used (e.g. *CyclicJobInformationType*).

NOTE: A second subtype *ContinuousJobInformationType* will be added in the future.

15.2.2 JobName

The *JobName Property* represents the name of the job.

15.2.3 JobDescription

The *JobDescription Property* represents the description of the job.

15.2.4 CustomerName

The *CustomerName Property* represents the name of the customer for that the job is produced.

15.2.5 ProductionDatasetName

The *ProductionDatasetName Property* represents the name of the production dataset which is needed for the job.

15.2.6 ProductionDatasetDescription

The *ProductionDatasetDescription Property* includes an additional description of the production dataset which is needed for the job.

15.2.7 Material

The *Material Property* is an Array of material names used for the job.

15.2.8 ProductName

The *ProductName Property* is an Array of product names produced by the job. (More than one possible with multiple cavities)

15.2.9 ProductDescription

The *ProductDescription Property* is an Array of descriptions of the products produced by the job.

15.2.10 ContinueAtJobEnd

The *ContinueAtJobEnd Property* indicates if the machine continues the production even if the nominal output has been reached.

15.2.11 CyclicJobInformationType

Additional information on a job for cyclic production (e.g. injection moulding) are stored in the *CyclicJobInformationType*. It extends the *JobInformationType.* It is formally defined in Table 69.

Table 69 – CyclicJobInformationType Definition

Attribute	**Value**				
BrowseName	CyclicJobInformationType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *JobInformationType*					
HasProperty	Variable	NominalParts	UInt64	PropertyType	M, RW
HasProperty	Variable	NominalBoxParts	UInt64	PropertyType	O, RW
HasProperty	Variable	ExpectedCycleTime	Duration	PropertyType	O, RW
HasProperty	Variable	MouldId	String	PropertyType	O, R
HasProperty	Variable	NumCavities	UInt32	PropertyType	O, R
HasComponent	Method	SetCyclicJobData			M
GeneratesEvent	ObjectType	RequestCyclicJobWriteEventType	Defined in 15.2.11.7		

15.2.11.1 NominalParts

The *NominalParts Property* indicates the total number (sum of all cavities) of parts that shall be produced by the job.

15.2.11.2 NominalBoxParts

In some productions, the parts are placed in several boxes. The *NominalBoxParts Property* indicates the number of parts that shall be put into one box.

NOTE: A "box" can be any kind of container like stacking box, lattice box, bag… .

15.2.11.3 ExpectedCycleTime

The *ExpectedCycleTime Property* indicates which cycle time is calculated for the job.

15.2.11.4 MouldId

The *MouldId Property* represents the Id of the *Mould* used for the job.

NOTE: Although a machine can be equipped with several moulds, one job is always related to a single mould. This is why *MouldId* is not an array here.

15.2.11.5 NumCavities

The *NumCavities Property* indicates the number of cavities in the *Mould* used for production.

15.2.11.6 SetCyclicJobData

With this *Method* the MES sets the job data for cyclic jobs.

Input arguments are all *Properties* of the *CyclicJobInformationType*.

Signature

```
SetCyclicJobData (
    [in]  String          JobName
    [in]  String          JobDescription
    [in]  String          CustomerName
    [in]  String          ProductionDatasetName
    [in]  String          ProductionDatasetDescription
    [in]  String[]        Material
    [in]  String[]        ProductName
    [in]  String[]        ProductDescription
    [in]  Boolean         ContinueAtJobEnd
    [in]  UInt64          NominalParts
    [in]  UInt64          NominalBoxParts
    [in]  Duration        ExpectedCycleTime
    [in]  String          MouldId
    [in]  UInt32          NumCavities);
```

When optional *Properties* in *CyclicJobInformationType* (e.g. *NominalBoxParts*) are not used, the arguments have be empty or equal to zero.

Table 70 – SetCyclicJobData Method AddressSpace Definition

Attribute	**Value**				
BrowseName	SetCyclicJobData				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory

15.2.11.7 RequestCyclicJobWrite

The *Event RequestCyclicJobWriteEventType* is used to initiate a call of the *SetCyclicJobData Method* by the client.

Table 71 – RequestCyclicJobWriteEventType Definition

Attribute	**Value**				
BrowseName	RequestCyclicJobWriteEventType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseEventType* defined in OPC UA Part 5					
HasProperty	Variable	JobName	String	PropertyType	M

15.3 Job Lists

The jobs for a machine can be planned in an MES. This clause defines methods and events to request and send a list of planned jobs.

15.3.1 SendJobList

This *Method* is used to send a list of jobs available on the client to the server. The server shall support to receive at least 10 jobs.

Signature

```
SendJobList (
    [in] JobListElementType[]      JobList);
```

Table 72 – SendJobList Method Arguments

Argument	Description
JobList	Array of JobInformationType describing the available jobs in the client.

Table 73 – SendJobList Method AddressSpace Definition

Attribute	Value				
BrowseName	SendJobList				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory

It is possible to call the *Method* without an *InputArgument* (length of the array *JobList* is zero) as an answer to the *Event RequestJobList* (see below), if no planned job is available.

The *JobListElementType* is defined in Table 74

Table 74 – JobListElementType Definition

Name	Type
JobListElementType	structure
JobName	String
JobDescription	String
JobClassification	String
CustomerName	String
ProductionDatasetName	String
ProductionDatasetDescription	String
Material	String[]
ProductName	String[]
ProductDescription	String[]
JobPriority	String
PlannedStart	DateTime
PlannedProductionTime	Duration
LatestEnd	DateTime

The variables *JobClassification*, *JobPriority*, *PlannedStart*, *PlannedProductionTime* and *LatestEnd* are additional information only for the operator to be shown on the machine control. This may help the operator to decide which jobs to download/activate. *PlannedStart* and *LatestEnd* shall be given in UTC time.
The *JobClassification* can e.g. be used to present a maintenance job with planned time. The possible values are user dependent and not standardized by this specification.

15.3.2 RequestJobList

The instance of *JobsType* can fire an *Event* of *RequestJobListEventType* (without parameters) to initiate a call of *SendJobList* by the client. This can e.g. be triggered by an operator who wants to see the planned jobs for his machine.

Table 75 – RequestJobListEventType Definition

Attribute	Value				
BrowseName	RequestJobListEventType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseEventType* defined in OPC UA Part 5					

15.3.3 SendCyclicJobList

This *Method* is used instead of *SendJobList* in the case of cyclic production and includes the input parameters *ExpectedCycleTime*, *NominalParts* and *NominalBoxParts*.

Signature

```
SendCyclicJobList (
    [in] CyclicJobListElementType[]    JobList);
```

Table 76 – SendCyclicJobList Method Arguments

Argument	Description
JobList	Array of CyclicJobInformationType describing the available cyclic jobs in the client.

Table 77 – SendCyclicJobList Method AddressSpace Definition

Attribute	Value				
BrowseName	SendCyclicJobList				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory

It is possible to call the *Method* without an *InputArgument* (length of the array *JobList* is zero) as an answer to the *Event RequestCyclicJobList* (see below), if no planned job is available.

The *CyclicJobListElementType* is a subtype of *JobListElementType* and adds the elements defined in Table 78.

Table 78 – CyclicJobListElementType Definition (subtype of JobListElementType)

Name	Type
CyclicJobListElementType	structure
NominalParts	UInt64
NominalBoxParts	UInt64
ExpectedCycleTime	Duration
MouldId	String
NumCavities	UInt32

NominalBoxParts, MouldId and *NumCavities* may be empty or zero if not used.

15.3.4 RequestCyclicJobList

The instance of *JobsType* can fire an *Event* of *RequestCyclicJobListEventType* (without parameters) to initiate a call of *SendCyclicJobList* by the client. This can e.g. be triggered by an operator who wants to see the planned jobs for his machine.

Table 79 – RequestCyclicJobListEventType Definition

Attribute	Value				
BrowseName	RequestCyclicJobListEventType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseEventType* defined in OPC UA Part 5					

15.4 ActiveJobValuesType

This *ObjectType* represents values of the active job. It is formally defined in Table 80.

Table 80 – ActiveJobValuesType Definition

Attribute	Value				
BrowseName	ActiveJobValuesType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType* defined in OPC UA Part 5					
HasProperty	Variable	JobStatus	JobStatusEnumeration	PropertyType	M, R
HasComponent	Method	StartJob			M
HasComponent	Method	InterruptJob			M
HasComponent	Method	FinishJob			M
HasProperty	Variable	CurrentLotName	String	PropertyType	M, RW
HasProperty	Variable	BoxId	String	PropertyType	O, RW
HasSubtype	ObjectType	ActiveCyclicJobValuesType			

NOTE: A second subtype *ActiveContinuousJobValuesType* will be added in the future.

15.4.1 JobStatus

The *JobStatus Property* represents the current status of the job.

Table 81 – JobStatusEnumeration Definition

Value	Description
OTHER_0	This state is used if none of the other states below apply. Set by operator.
TRANSFERRED_ASSIGNED_1	The job has been transferred to the machine and assigned as current job.
SET_UP_ACTIVE_2	The operator prepares the machine for the job.
SET_UP_INTERRUPTED_3	The operator has interrupted but not finished the preparation of the machine for the job.
SET_UP_FINISHED_4	The operator has finished the preparation of the machine for the job.
START_UP_ACTIVE_5	The operator is setting the machine in the start-up phase.
JOB_IN_PRODUCTION_6	The machine is producing parts/products for the job.
JOB_INTERRUPTED_7	The job is interrupted. The nominal output is not reached.
JOB_FINISHED_8	Nominal output reached.
TEAR_DOWN_ACTIVE_9	The operator tears the machine down.
TEAR_DOWN_INTERRUPTED_10	Tear-down is interrupted but not finished.
TEAR_DOWN_FINISHED_11	Tear-down is finished.

The *JobStatus* is set by the machine/operator. This can be triggered by the methods Start*Job*, *InterruptJob* and *FinishJob*. The selection of the value JOB_IN_PRODUCTION_6 can be prevented by the client by setting *ProductionReleased* to FALSE (see 11.7.2).

15.4.2 StartJob

With this *Method* the client (e.g. MES) request the machine to change the *JobStatus* to JOB_IN_PRODUCTION_6. The following produced parts/products are counted for the job.

NOTE: The method does not change the *MachineMode.*

Signature

```
StartJob ();
```

The method has no *Input*- or *OutputArguments*.

Table 82 – StartJob Method AddressSpace Definition

Attribute	Value				
BrowseName	StartJob				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**

15.4.3 InterruptJob

With this *Method* the client (e.g. MES) requests the machine to change the *JobStatus* to JOB_INTERRUPTED_7. The following produced parts/products are not counted for the job.

NOTE: It is not fixed by this specification if the machine stops or continuous running.

Signature

```
InterruptJob ();
```

The method has no *Input*- or *OutputArguments*.

Table 83 – InterruptJob Method AddressSpace Definition

Attribute	Value				
BrowseName	InterruptJob				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**

15.4.4 FinishJob

With this *Method* the client (e.g. MES) requests the machine to change the *JobStatus* to JOB_FINISHED_8. If *ContinueAtJobEnd* (see 15.2.10) is FALSE, the machine stops. Otherwise a message to the operator shall be shown on the machine.

Signature

```
FinishJob ();
```

The method has no *Input*- or *OutputArguments*.

Table 84 – FinishJob Method AddressSpace Definition

Attribute	Value				
BrowseName	FinishJob				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**

15.4.5 CurrentLotName

The *CurrentLotName Property* represents the current production lot. Although the modelling rule is mandatory, the string can be empty.

15.4.6 BoxId

The *BoxId Property* represents the Id of the box in which the current production is put in.

NOTE: A "box" can be any kind of container like stacking box, lattice box, bag… .

15.4.7 ActiveCyclicJobValuesType

Additional information on the running job for cyclic production (e.g. injection moulding) is stored in the *ActiveCyclicJobValuesType*. It extends the *ActiveJobValuesType*. It is formally defined in Table 85.

Table 85 – ActiveCyclicJobValuesType Definition

Attribute	**Value**				
BrowseName	ActiveCyclicJobValuesType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *ActiveJobValuesType*					
HasComponent	Variable	JobCycleCounter	UInt64	BaseDataVariableType	M, R
HasComponent	Variable	BoxCycleCounter	UInt64	BaseDataVariableType	O, R
HasComponent	Variable	MachineCycleCounter	UInt64	BaseDataVariableType	O, R
HasComponent	Variable	LastCycleTime	Duration	BaseDataVariableType	M, R
HasComponent	Variable	AverageCycleTime	Duration	BaseDataVariableType	O, R
HasComponent	Variable	JobPartsCounter	UInt64	BaseDataVariableType	M, R
HasComponent	Variable	JobGoodPartsCounter	UInt64	BaseDataVariableType	M, R
HasComponent	Variable	JobBadPartsCounter	UInt64	BaseDataVariableType	M, R
HasComponent	Variable	JobTestSamplesCounter	UInt64	BaseDataVariableType	M, R
HasComponent	Variable	BoxPartsCounter	UInt64	BaseDataVariableType	O, R
HasComponent	Variable	BoxGoodPartsCounter	UInt64	BaseDataVariableType	O, R
HasComponent	Variable	BoxBadPartsCounter	UInt64	BaseDataVariableType	O, R
HasComponent	Variable	BoxTestSamplesCounter	UInt64	BaseDataVariableType	O, R
HasComponent	Variable	LastPartId	String[]	BaseDataVariableType	O, R
HasComponent	Method	StopAtCycleEnd			M
HasComponent	Method	ResetJobCounters			M
HasComponent	Method	ResetBoxCounters			O
HasComponent	Method	ResetAverageCycleTime			O

15.4.7.1 JobCycleCounter

The *JobCycleCounter Variable* represents the number of finished cycles in the job.

15.4.7.2 BoxCycleCounter

The *BoxCycleCounter Variable* represents the number of finished cycles for the current box.

15.4.7.3 MachineCycleCounter

The *MachineCycleCounter Variable* represents the number of finished cycles in the machine life time.

15.4.7.4 LastCycleTime

The *LastCycleTime Variable* represents the cycle time (duration in milliseconds) of the recently finished cycle.

15.4.7.5 AverageCycleTime

The *AverageCycleTime Variable* represents the average cycle time. The calculation starts from the last calling of *ResetActiveJobAverageCycleTime*.

15.4.7.6 JobPartsCounter, JobGoodPartsCounter, JobBadPartsCounter, JobTestSamplesCounter

These *Variables* represent the total number of produced parts and the numbers of good, bad and test sample parts in the current job.

15.4.7.7 BoxPartsCounter, BoxGoodPartsCounter, BoxBadPartsCounter, BoxTestSamplesCounter

These *Variables* represent the total number of produced parts and the numbers of good, bad and test sample parts in the current box.

15.4.7.8 LastPartId

The *LastPartId Variable* is an array and represents the Ids of the parts produced in the recently finished cycle.

15.4.7.9 StopAtCycleEnd

With this *Method,* the MES directs the machine to stop at the end of the current cycle.

Signature

```
StopAtCycleEnd ();
```

The method has no *Input*- or *OutputArguments*.

Table 86 – StopAtCycleEndMethod AddressSpace Definition

Attribute	Value				
BrowseName	StopAtCycleEnd				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**

15.4.7.10 ResetJobCounters

Method for setting the cycle and parts counters for the job to 0.

Signature

```
ResetJobCounters ();
```

The method has no *Input*- or *OutputArguments*.

Table 87 – ResetJobCounters Method AddressSpace Definition

Attribute	Value				
BrowseName	ResetJobCounters				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**

15.4.7.11 ResetBoxCounters

Method for setting the cycle and parts counters for the current box to 0.

Signature

```
ResetBoxCounters ();
```

The method has no *Input*- or *OutputArguments*.

Table 88 – ResetBoxCounters Method AddressSpace Definition

Attribute	Value				
BrowseName	ResetBoxCounters				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**

15.4.7.12 ResetAverageCycleTime

This *Method* initiates a new calculation of the average cycle time for the job.

Signature

```
ResetAverageCycleTime ();
```

The method has no *Input*- or *OutputArguments*.

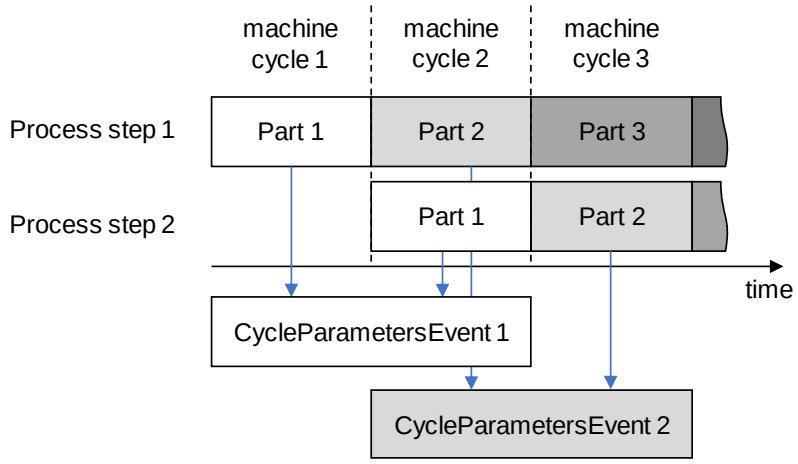
Table 89 – ResetAverageCycleTime Method AddressSpace Definition

Attribute	Value				
BrowseName	ResetAverageCycleTime				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**

16 CycleParametersEventType

The *CycleParametersEventType* represents information on a production cycle. It is fired after each finished cycle of the machine (this can be different from quality checks by downstream equipment).

NOTE: If parts a produced in two or more steps, the cycle parameters shall be related to the finished parts. E.g. for a two-step production with two moulds, the cycle parameters after a finished cycle include the temperatures/pressures/… values from the previous machine cycle for the first production step and the values from the currently finished machine cycle for the second step:



The *EventSource* is the root node of the interface (e.g. instance of *IMM_MES_InterfaceType* for EUROMAP 77).

NOTE: There is no *CycleParameters Object* which is directly accessible via data access, because this could lead to inconsistent data.

The *CycleParametersEventType* is formally defined in Table 90.

Table 90 – CycleParametersEventType Definition

Attribute	**Value**				
BrowseName	CycleParametersEventType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseEventType* defined in OPC UA Part 5					
HasProperty	Variable	JobName	String	PropertyType	M
HasProperty	Variable	JobStatus	JobStatusEnumeration	PropertyType	M
HasProperty	Variable	CurrentLotName	String	PropertyType	M
HasProperty	Variable	BoxId	String	PropertyType	O
HasProperty	Variable	JobCycleCounter	UInt64	PropertyType	M
HasProperty	Variable	BoxCycleCounter	UInt64	PropertyType	O
HasProperty	Variable	MachineCycleCounter	UInt64	PropertyType	O
HasProperty	Variable	CycleTime	Duration	PropertyType	M
HasProperty	Variable	AverageCycleTime	Duration	PropertyType	O
HasProperty	Variable	JobPartsCounter	UInt64	PropertyType	M
HasProperty	Variable	JobGoodPartsCounter	UInt64	PropertyType	M
HasProperty	Variable	JobBadPartsCounter	UInt64	PropertyType	M
HasProperty	Variable	JobTestSamplesCounter	UInt64	PropertyType	M
HasProperty	Variable	BoxPartsCounter	UInt64	PropertyType	O
HasProperty	Variable	BoxGoodPartsCounter	UInt64	PropertyType	O
HasProperty	Variable	BoxBadPartsCounter	UInt64	PropertyType	O
HasProperty	Variable	BoxTestSamplesCounter	UInt64	PropertyType	O
HasProperty	Variable	CycleQuality	CycleQualityEnumeration	PropertyType	M
HasProperty	Variable	CavityCycleQuality	CavityCycleQualityEnumeration[]	PropertyType	O
HasProperty	Variable	PartId	String []	PropertyType	O

The *CycleParametersEventType* can be extended to include additional information. For this, a new subtype shall be derived in the OPC Server of the machine (Note: the namespace of the derived *Type* is then the Local Server URI with namespace index 1 or a vendor specific namespace with vendor specific index). The EUROMAP recommendations provide some standardized *ObjectTypes* that can be included (e.g. *MouldCycleParametersType* as defined in 16.14, *InjectionUnitCycleParametersType* as defined in EUROMAP 77).

16.1 JobName

The *JobName Property* represents the name of the job.

16.2 JobStatus

The *JobStatus Property* represents the current status of the job (see15.4.1).

16.3 CurrentLotName

The *CurrentLotName Property* represents the current production lot.

16.4 BoxId

The *BoxId Property* represents the Id of the box in which the current production is put in.

16.5 CycleCounter

The *CycleCounter Property* represents the number of the cycle in the job.

16.6 MachineCycleCounter

The *MachineCycleCounter Property* represents the number of finished cycles in the machine life time.

16.7 CycleTime

The *CycleTime Variable* represents the cycle time.

16.8 AverageCycleTime

The *AverageCycleTime Variable* represents the average cycle time. The calculation starts from the last calling of *ResetActiveJobAverageCycleTime*.

16.9 JobPartsCounter, JobGoodPartsCounter, JobBadPartsCounter, JobTestSamplesCounter

These *Properties* represent the total number of produced parts and the numbers of good, bad and test sample parts in the current job.

16.10 BoxPartsCounter, BoxGoodPartsCounter, BoxBadPartsCounter, BoxTestSamplesCounter

These *Properties* represent the total number of produced parts and the numbers of good, bad and test sample parts in the current box.

16.11 CycleQuality

The *CycleQuality Property* gives information on the quality of the whole cycle. The *CycleQualityEnumeration* is defined in Table 91.

Table 91 – CycleQualityEnumeration Definition

Value	Description
GOOD_CYCLE_0	The machine has detected no failures during the cycle and the part quality (for all cavities) is assumed as good.
BAD_CYCLE_1	The quality of the part(s) is assumed as bad. If the machine is able to evaluate the cycle quality of each cavity, detailed information can be given in the *CavityCycleQuality Property*. Nevertheless, already a bad part in only one cavity leads to a BAD_CYLE value for the *CycleQuality*.
TEST_SAMPLE_CYCLE_2	A cycle is separated as a test sample. Triggered by operator or MES.
FAILED_CYCLE_3	The machine has detected failures during the cycle and the part quality is assumed as bad. Further information is provided by the *MessageCondition* fired in this case.

16.12 CavityCycleQuality

The *CavityCycleQuality Property* gives information on the quality of the cycle for each cavity if the machine is able to evaluate this.

Table 92 – CavityCycleQualityEnumeration Definition

Value	Description
NO_PART_0	There is no part in cavity.
GOOD_PART_1	The machine has detected no failures during the cycle for this cavity and the part quality is assumed as good.
BAD_PART_2	The machine has detected failures during the cycle for the cavity and the part quality is assumed as bad.
REWORK_3	The machine has detected failures during the cycle for the cavity which might be fixed by reworking the part

16.13 PartId

The *PartId Property* is an array and represents the Ids of the parts produced in the cycle.

NOTE: The Id(s) may be generated in the machine or coming from outside (e.g. from MES or labelling machine).

16.14 MouldCycleParametersType

The *MouldCycleParametersType* represents information on the production cycle related to a mould (mainly temperatures). As the number of moulds and the related temperature zones are varying, the *EventType* for the cycle parameters has to be derived from *CycleParametersEventType* by the OPC Server of the machine (Note: the namespace of the derived *Type* is then the Local Server URI with namespace index 1 or a vendor specific namespace with vendor specific index). When the *MouldCycleParametersType* is used, the *BrowseNames* of the additional objects shall be “MouldCycleParameters_<Nr>” (starting with 1)

Table 93 – Example of an event type derived from CycleParametersEventType with two moulds

Attribute	Value				
BrowseName	ExampleCycleParametersEventType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *CycleParametersEventType*					
HasComponent	Object	MouldCycleParameters_1		Example1MouldCycle ParametersType	M
HasComponent	Object	MouldCycleParameters_2		Example2MouldCycle ParametersType	M

The Types *Example1MouldCycleParametersType* and *Example2MouldCycleParametersType* used in the example are subtypes of the *MouldCycleParametersType* which is formally defined in Table 94.

Table 94 – MouldCycleParametersType Definition

Attribute	Value				
BrowseName	MouldCycleParametersType				
IsAbstract	True				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType* defined in OPC UA Part 5					
HasProperty	Variable	Index	UInt32	PropertyType	M, R

The *MouldCycleParametersType* is abstract and the OPC server of the machine shall create a derived type with the additional instances of *TemperatureZoneCycleParametersType* for the temperature zones of the mould. The *BrowseNames* of the objects shall be “MouldTemperatureZone_<Nr>” (starting with 1 for each mould).

Table 95 – Example of an object type derived from MouldCycleParametersType

Attribute	Value				
BrowseName	Example1MouldCycleParametersType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *MouldCycleParametersType*					
HasComponent	Object	MouldTemperatureZone_1		TemperatureZoneCycleParametersType	M
HasComponent	Object	MouldTemperatureZone_2		TemperatureZoneCycleParametersType	M
HasComponent	Object	MouldTemperatureZone_3		TemperatureZoneCycleParametersType	M
HasComponent	Object	MouldTemperatureZone_4		TemperatureZoneCycleParametersType	M

16.15 TemperatureZoneCycleParametersType

This *ObjectType* is used for the temperatures in barrel and mould zones. When instances are created, the *BrowseNames* shall be “BarrelTemperatureZoneCycleParameters_<Nr>”, respectively “MouldTemperatureZone_<Nr>” (starting with 1).

The *TemperatureZoneCycleParametersEventType* is formally defined in Table 96.

Table 96 – TemperatureZoneCycleParametersType Definition

Attribute	Value				
BrowseName	TemperatureZoneCycleParametersType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType* defined in OPC UA Part 5					
HasProperty	Variable	Index	UInt32	PropertyType	M
HasProperty	Variable	Name	String	PropertyType	M
HasProperty	Variable	Classification	TemperatureZone ClassificationEnumeration	PropertyType	O, R
HasComponent	Variable	ActualTemperature	Double	AnalogItemType	M

The *Index Property* gives the number of the zone.

The *Name Property* represents the name of the zone.

The *Classification Property* represents the type of the zone. The *TemperatureZoneClassificationEnumeration* is defined in 14.5.

The *ActualTemperature Variable* represents the current temperature of the zone.

17 ProductionDatasetManagementType

17.1 General

Production datasets are used for the configuration of machines. They are stored in files which can be exchanged over the network (e.g. between a machine and MES). This chapter defines methods and events for the exchange of lists of available production datasets and for the transfer of the files themselves.

NOTE: Transferred production datasets are always complete and consistent (includes data for the core machine and connected peripheral equipment, e.g. robot). As the production dataset files themselves are not standardized, a set of metadata is included in the *ProductionDatasetInformationType* (see 17.4.4).

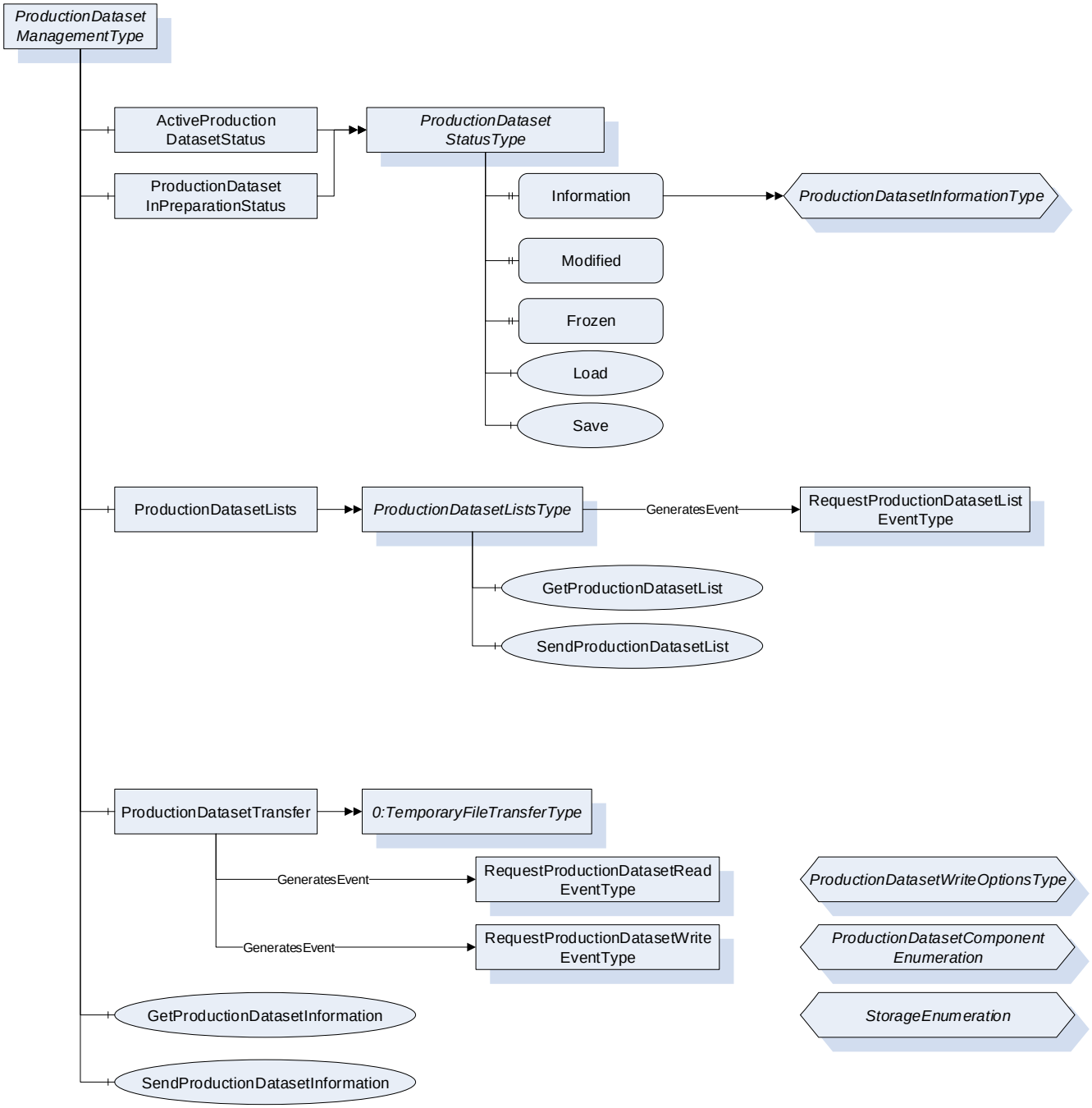


Figure 11 – *ProductionDatasetManagementType* Overview

17.2 ProductionDatasetManagementType Definition

This *ObjectType* is a container for the functionalities related to the listing and exchange of production datasets. It is formally defined in Table 97.

Table 97 – ProductionDatasetManagementType Definition

Attribute	Value				
BrowseName	ProductionDatasetManagementType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType* defined in OPC UA Part 5					
HasComponent	Object	ActiveProductionDatasetStatus		ProductionDatasetStatusType	M
HasComponent	Object	ProductionDatasetInPreparationStatus		ProductionDatasetStatusType	O
HasComponent	Object	ProductionDatasetLists		ProductionDatasetListsType	O
HasComponent	Object	ProductionDatasetTransfer		TemporaryFileTransferType	M
HasComponent	Method	GetProductionDatasetInformation			O
HasComponent	Method	SendProductionDatasetInformation			O

17.3 ProductionDatasetStatusType

17.3.1 ActiveProductionDatasetStatus, ProductionDatasetInPreparationStatus

This *Object* represents the status of the production dataset which is active in the control system of the machine or in preparation (in analogy to *ActiveJob* and *JobInPreparation*). The *ProductionDatasetStatusType* is formally defined in Table 98.

NOTE: Production datasets are manufacturer specific files which contain settings of production parameters for the machine and for connected peripheral devices.

Table 98 – ProductionDatasetStatusType Definition

Attribute	Value				
BrowseName	ProductionDatasetStatusType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType* defined in OPC UA Part 5					
HasProperty	Variable	Information	ProductionDataset InformationType	PropertyType	M, R
HasProperty	Variable	Modified	Boolean	PropertyType	O, R
HasProperty	Variable	Frozen	Boolean	PropertyType	O, RW
HasComponent	Method	Load			O
HasComponent	Method	Save			O

17.3.2 Information

The *Information Property* represents a set of information on the production dataset.

17.3.3 Modified

The *Modified Property* informs if the production dataset has been changed after the last storage.

NOTE: This information is only valid for the machine directly connected to the client. If the dataset also includes parameters for peripheral devices to that machine, changes in the peripheral devices might not be recognized.

17.3.4 Frozen

The *Frozen Property* changes in the production dataset are not allowed. If TRUE, no changes on the machine in the production dataset (change of process parameters) are allowed.

NOTE: This information is only valid for the machine directly connected to the client. If the dataset also includes parameters for peripheral devices to that machine, changes in the peripheral devices might still be possible.

17.3.5 Load

The *Method Load* loads a production dataset from the file system of the machine to the control of the machine. As production datasets can contain parameter settings not only for the machine itself but also for peripheral equipment (e.g. robots/handling devices), the parts of the production dataset which shall be activated can be chosen.

Signature

```
Load (
   [in]  String                                    Name
   [in]  ProductionDatasetComponentEnumeration[]   Components);
```

Table 99 – Load Method Arguments

Argument	Description
Name	Name of the production dataset that should be loaded.
Components	Indication which parts of the production dataset shall be activated. If *ProductionDatasetComponent* is not given (array length 0) then the complete Production dataset is activated.

Table 100 – Load Method AddressSpace Definition

Attribute	Value				
BrowseName	Load				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory

The *ProductionDatasetComponentEnumeration* is defined in Table 110.

17.3.6 Save

The *Method Save* stores a production dataset from the control of the machine to the file system of the machine.

Signature

```
Save (
   [in]  String      Name);
```

Table 101 – Save Method Arguments

Argument	Description
Name	Name under which the production dataset that should be stored in the file system.

Table 102 – Save Method AddressSpace Definition

Attribute	Value				
BrowseName	Save				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory

17.4 ProductionDatasetLists

The *Object ProductionDatasetLists* is used to exchange information on the available production datasets on client and server.

Table 103 – ProductionDatasetListsType Definition

Attribute	Value				
BrowseName	ProductionDatasetListsType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType* defined in OPC UA Part 5					
HasComponent	Method	GetProductionDatasetList			M
HasComponent	Method	SendProductionDatasetList			M
GeneratesEvent	ObjectType	RequestProductionDatasetListEventType	Defined in 17.4.3		

The *Object ProductionDatasetLists* can fire an *Event RequestProductionDatasetList* to initiate a call of *SendProductionDataList* by the MES.

17.4.1 GetProductionDatasetList

This *Method* is used to read a list from the server which production datasets are available on the machine's file system (e.g. for a check before an activation or transfer of a production dataset is initiated). The *NameFilter* and *MouldId* can be used to reduce the length of the list or for a targeted search.

Signature

```
GetProductionDatasetList (
    [in]  String                                NameFilter
    [in]  String                                MouldId
    [out] ProductionDatasetInformationType[]    ProductionDatasetList);
```

Table 104 – GetProductionDatasetList Method Arguments

Argument	Description
NameFilter	The Filter can be used to reduce the length of the list or for a targeted search. The wildcards "*" and "?" may be used. *NameFilter* = "" requests a lists of all available production datasets, *NameFilter* = "300" requests only information on the production dataset with *ProductionDatasetName* "300" (if available), *NameFilter* = "3*" requests information on production datasets with a *ProductionDatasetName* starting with "3" (e.g. "300", "301", …).
MouldId	If *MouldId*<>"" only production datasets for the given *MouldId* are requested.
ProductionDatasetList	Array of ProductionDatasetInformationType (see 17.4.4) describing the available production datasets on the server.

Table 105 – GetProductionDatasetList Method AddressSpace Definition

Attribute	Value				
BrowseName	GetProductionDatasetList				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Mandatory

17.4.2 SendProductionDatasetList

This *Method* is used to send a list of production datasets available on the client to the server.

Signature

```
SendProductionDatasetList (
    [in]  ProductionDatasetInformationType[]    ProductionDatasetList);
```

Table 106 – SendProductionDatasetList Method Arguments

Argument	Description
ProductionDatasetList	Array of ProductionDatasetInformationType describing the available production datasets in the MES.

Table 107 – SendProductionDatasetList Method AddressSpace Definition

Attribute	Value				
BrowseName	SendProductionDatasetList				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory

It is possible to call the *Method* without an *InputArgument* (length of the array *ProductionDatasetList* is zero) as an answer to the *Event RequestProductionDatasetList* (see below) if no production dataset which fits the *NameFilter* and/or *MouldId* transferred with the event is available.

17.4.3 RequestProductionDatasetList

The *Object ProductionDatasetLists* can fire an *Event RequestProductionDatasetListEventType* to initiate a call of *SendProductionDatasetList* by the client. This can for example be triggered by an operator who wants to load a production dataset not existing on the server.

Table 108 – RequestProductionDatasetListEventType Definition

Attribute	Value				
BrowseName	RequestProductionDatasetListEventType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseEventType* defined in OPC UA Part 5					
HasProperty	Variable	NameFilter	String	PropertyType	M
HasProperty	Variable	MouldId	String	PropertyType	M

The *MouldFilter* and *MouldId* parameters which can be used to reduce the length of the list or for a targeted search is used in the same way as in the *Method GetProductionDatasetList* defined in 17.4.1.

17.4.4 ProductionDatasetInformationType

This structure provides information on a production dataset. It does not contain the production dataset file itself.

Table 109 – ProductionDatasetInformationType Definition

Name	Type	Description
ProductionDatasetInformationType	structure	
Name	String	Name of the production dataset (=identifier).
Description	String	Additional description of the production dataset.
MESId	String	Id of the production dataset file assigned by MES.
CreationTimestamp	DateTime	Time when the production dataset was originally created or saved with a new name (in UTC time).
LastModificationTimestamp	DateTime	Time of the last modification of the production dataset (in UTC time).
LastSaveTimestamp	DateTime	Time when the production dataset was saved to the file system of the machine (in UTC time).
UserName	String	Name of the user who has made the last parameter change on the machine.
Components	ProductionDataset ComponentEnumeration[]	Informs for which machines information is included in the file. The *ProductionDatasetComponentEnumeration* is defined in Table 110. If the production dataset contains information for two machines of the same types, the enumeration value is repeated. Example: The production dataset contains information for 1 injection moulding machine and 2 robots: Components = [0 1 1].
Manufacturer	String	These Properties are representing the values stored in the *MachineInformation* at the time the production dataset file is created.
SerialNumber	String	
Model	String	
ControllerName	String	
UserMachineName	String	These Properties are taken from the *MachineConfiguration* at the time the production dataset file is created.
LocationName	String	
ProductName	String[]	These Properties are in analogy with the Properties in the *JobInformationType*. *MouldId* and *NumCavities* are optional in the derived *CyclicJobInformationType*, so here empty strings / value zero are possible.
MouldId	String	
NumCavities	UInt32	

NOTE: This meta data shall also be included in the production dataset file itself.

Table 110 – ProductionDatasetComponentEnumeration

Value	Name
IMM_0	Injection moulding machine
ROBOT_1	Robot or handling device

NOTE: The *ProductionDatasetComponentEnumeration* will be extended when further OPC UA models for other machine types are developed.

17.5 ProductionDatasetTransfer

17.5.1 General

OPC UA Part 5 defines a *TemporaryFileTransferType* for the representation of file transfers. This Type is used for the transfer of production datasets.

17.5.2 GenerateOptions in GenerateFileForRead

For the *GenerateOptions* in the *Method GenerateFileForRead* the *DataType ProductionDatasetReadOptionsType* as defined below shall be used.

Table 111 – ProductionDatasetReadOptionsType Definition

Name	Type	Description
ProductionDatasetReadOptionsType	structure	
Storage	StorageEnumeration	Indication from where the production dataset is read. Enumeration of Type *StorageEnumeration.* Although StorageEnumeration is defined as a Mask, here only the values 1, 2 and 4 are allowed (no combination).
Name	String	Name of the production dataset that should be transferred from the server to the client. This parameter is only relevant, if *Storage* is 4 (FILE_SYSTEM). In other cases, it shall be an empty string.

17.5.3 GenerateOptions in GenerateFileForWrite

For the *GenerateOptions* in the *Method GenerateFileForWrite* the *DataType ProductionDatasetWriteOptionsType* as defined below shall be used.

Table 112 – ProductionDatasetWriteOptionsType Definition

Name	Type	Description
ProductionDatasetWriteOptionsType	structure	
Storage	StorageEnumeration	Indication where the production dataset is written to. Enumeration of Type *StorageEnumeration.*
Name	String	Name of the production dataset that should be transferred from the client to the server.
Components	ProductionDatasetComponent Enumeration[]	Array which indicates which parts of the production dataset shall be activated in the machine control after writing. Only valid if *Storage* is PRODUCTION_1 or PREPARATION_2. Array of Enumeration of Type *ProductionDatasetComponentEnumeration.* If Components has the array length 0 then complete production dataset is activated.

Table 113 – StorageEnumeration Definition

Value	Description
PRODUCTION_1	The production dataset is written directly to the (active layer of the) control system of the machine.
PREPARATION_2	The production dataset is written to the preparation layer of the control system of the machine (if supported).
FILE_SYSTEM_4	The production dataset is written to the file system of the machine for later activation.

This *Enumeration* is defined as a *Mask*. With this, writing to several destinations with one *Method* call is possible (e.g. *StorageEnumeration* = 6 writes the production dataset to the file system and the preparation layer of the control system.

NOTE: It is possible that the machine does not support all storage options (e.g. only writing to file system allowed). In this case, the server will return the *StatusCode* Bad_InvalidArgument when calling the *GenerateFileForWrite Method* with a not supported value for *Storage*.

17.6 Events for ProductionDatasetTransfer

RequestProductionDatasetRead and *RequestProductionDatasetWrite* are *Events* to trigger a file transfer by the machine/server (e.g. initiated by the operator).

Table 114 – RequestProductionDatasetReadEventType Definition

Attribute	Value				
BrowseName	RequestProductionDatasetReadEventType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseEventType* defined in OPC UA Part 5					
HasProperty	Variable	Storage	StorageEnumeration	PropertyType	M
HasProperty	Variable	Name	String	PropertyType	M

Storage: Indication from where the dataset is read. Although *StorageEnumeration* is defined as a *Mask*, here only the values 1, 2 and 4 are allowed (no combination).
Name: Name of the production dataset that should be transferred from the server to the client. This parameter is only relevant, if *Storage* is 4 (FILE_SYSTEM).

Table 115 – RequestProductionDatasetWriteEventType Definition

Attribute	Value				
BrowseName	RequestProductionDatasetWriteEventType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseEventType* defined in OPC UA Part 5					
HasProperty	Variable	Storage	StorageEnumeration	PropertyType	M
HasProperty	Variable	Name	String	PropertyType	M
HasProperty	Variable	Components	ProductionDatasetComponent Enumeration[]	PropertyType	M

Name: Name of the production dataset that should be transferred from the client to the server.
Storage: Indication where the dataset is written to.
ProductionDatasetComponent: Array which indicates which *parts* of the production dataset shall be activated in the machine control after writing. Only valid if *Storage* is CONTROL_SYSTEM_1 or FILE_AND_CONTROL_SYSTEM_2. Array of *Enumeration* of *Type*. *ProductionDatasetComponentEnumeration*. If *ProductionDatasetComponent* has the array length 0 then complete production dataset is activated.

17.7 GetProductionDatasetInformation

This *Method* allows reading the description of a production dataset during the file transfer from the server to the client with *ProductionDatasetTransfer*. It may only be called between receiving the *fileHandle* generated by *GenerateFileForRead* and closing the file.

Signature

```
GetProductionDatasetInformation (
    [in]  UInt32                                fileHandle
    [out] ProductionDatasetInformationType      Information);
```

Table 116 – GetProductionDatasetInformation Method Arguments

Argument	Description
fileHandle	Value of *fileHandle* received by *GenerateFileForRead* in *ProductionDatasetTransfer*
Information	Description of the production dataset with ProductionDatasetInformationType (see 17.4.4)

Table 117 – GetProductionDatasetInformation Method AddressSpace Definition

Attribute	Value				
BrowseName	GetProductionDatasetInformation				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Mandatory

17.8 SendProductionDatasetInformation

This *Method* allows sending of the description of a production dataset during the file transfer from the client to the server with *ProductionDatasetTransfer*. It may only be called between receiving the *fileHandle* generated by *GenerateFileForWrite* and closing the file.

Signature

```
SendProductionDatasetInformation (
    [in]  UInt32                              fileHandle
    [in]  ProductionDatasetInformationType    Information);
```

Table 118 – GetProductionDatasetInformation Method Arguments

Argument	Description
fileHandle	Value of *fileHandle* received by *GenerateFileForRead* in *ProductionDatasetTransfer*
Information	Description of the production dataset with ProductionDatasetInformationType (see 17.4.4)

Table 119 – GetProductionDatasetInformation Method AddressSpace Definition

Attribute	Value				
BrowseName	GetProductionDatasetInformation				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory

18 IdentificationType

The *IdentificationType* represents general information about a machine. The information is fixed by the manufacturer and not changeable by the user. It is used instead of the *MachineInformationType* where the detailed information of the *MachineInformationType* is not needed e.g. in a horizonal communication between machines in a production line.

Table 120 – IdentificationType Definition

Attribute	**Value**				
BrowseName	IdentificationType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType*					
HasProperty	Variable	DeviceClass	String	PropertyType	M, R
HasProperty	Variable	Manufacturer	String	PropertyType	M, R
HasProperty	Variable	Model	String	PropertyType	M, R
HasProperty	Variable	SerialNumber	String	PropertyType	M, R
HasProperty	Variable	ArticleNumber	String	PropertyType	O, R
HasProperty	Variable	SoftwareRevision	String	PropertyType	O, R
HasProperty	Variable	YearOfConstruction	UInt16	PropertyType	O, R

18.1 DeviceClass

The *DeviceClass Property* indicates in which domain or for what purpose a certain device is used. The value is specified in the specific Companion Specification (e.g. "Injection Moulding Machine" for EUROMAP 77)

18.2 Manufacturer

The *Manufacturer Property* provides the name of the manufacturer of the machine (e.g. "Negri Bossi").

18.3 Model

The *Model Property* represents the name of the machine type (e.g. "KM 1000-2500", "Allrounder").

18.4 SerialNumber

The *SerialNumber Property* represents the serial number of the machine (unique ID given by the manufacturer, e.g. "1240114").

18.5 ArticleNumber

The *ArticleNumber Property* represents the article number of the machine (e.g. "123.4567.89").

18.6 SoftwareRevision

The *SoftwareRevision Property* represents the software version used in the control unit (e.g. "nb2001v11B030").

18.7 YearOfConstruction

The *YearOfConstruction Property* represents the year of construction of the machine (e.g. "2018").

19 MonitoredParameterType

The *MonitoredParameterType* is used for process parameters that are monitored by the client.

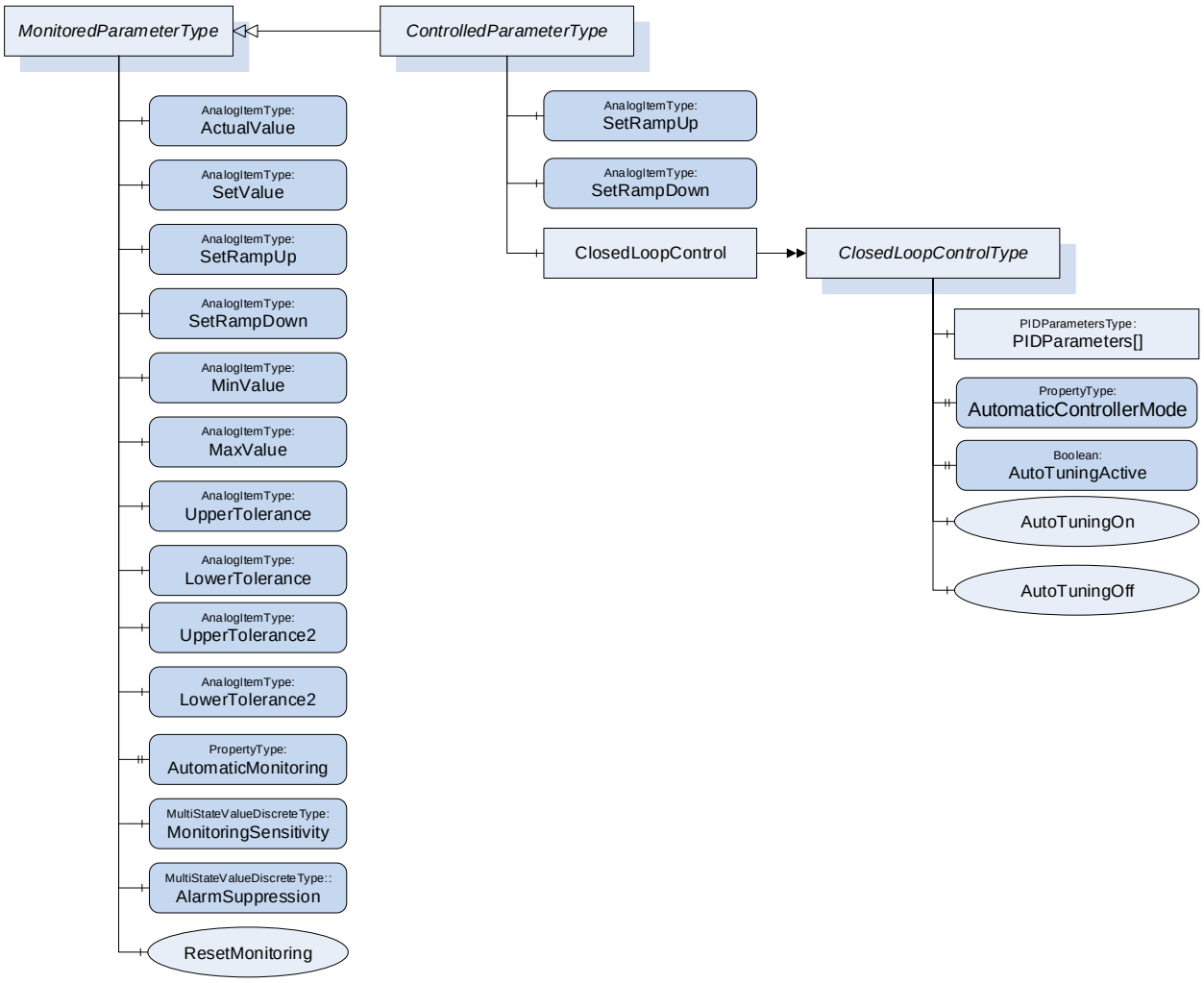


Figure 12 – MonitoredParameterType Overview

Table 121 – MonitoredParameterType Definition

Attribute	**Value**				
BrowseName	MonitoredParameterType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType*					
HasComponent	Variable	ActualValue	Double	AnalogItemType	M, R
HasComponent	Variable	SetValue	Double	AnalogItemType	O, RW
HasComponent	Variable	SetRampUp	Double	AnalogItemType	O, R
HasComponent	Variable	SetRampDown	Double	AnalogItemType	O, R
HasComponent	Variable	MinValue	Double	AnalogItemType	O, RW
HasComponent	Variable	MaxValue	Double	AnalogItemType	O, RW
HasComponent	Variable	UpperTolerance	Double	AnalogItemType	O, RW
HasComponent	Variable	LowerTolerance	Double	AnalogItemType	O, RW
HasComponent	Variable	UpperTolerance2	Double	AnalogItemType	O, RW
HasComponent	Variable	LowerTolerance2	Double	AnalogItemType	O, RW
HasProperty	Variable	AutomaticMonitoring	Boolean	PropertyType	O, RW
HasComponent	Variable	MonitoringSensitivity	UInt16	MultiStateValueDiscreteType	O, RW
HasComponent	Variable	AlarmSuppression	UInt16	MultiStateValueDiscreteType	O, RW
HasComponent	Method	ResetMonitoring			O
HasSubtype	ObjectType	ControlledParameterType	Defined in Clause 20		

19.1 ActualValue

Actual value of the monitored parameter (unit given in *AnalogItemType*).

19.2 SetValue

Set/nominal/target value of the monitored parameter. The value of this variable is writeable by the client. As the *MonitoredParameterType* is not used to control parameters, this is only for information/Process monitoring and not for changing setting on the device.

NOTE: For controlling parameters the ControlledParameterType is defined in Clause 20.

19.3 SetRampUp

Indication if a SetValue that is higher than the actual value shall be reached as fast as possible (*SetRampUp* = 0) or within a given value change per time (e.g. *SetRampUp* = 2 K/s).

19.4 SetRampDown

Indication if SetValue that is lower than the actual value shall be reached as fast as possible (*SetRampDown* = 0) or within a given value change per time (e.g. *SetRampDown* = 2 K/s).

NOTE: Always positive value.

19.5 UpperTolerance, LowerTolerance, UpperTolerance2, LowerTolerance2, MinValue, MaxValue,

These parameters are used to define limits for the monitored parameter. Exceeding the (relative) tolerance values creates a warning while exceeding the (absolute) Min/MaxValues leads to an alarm from type HelpOffNormalAlarmType and/or perhaps other actions on the machine (e.g. switching off the heating, stopping of movements) as defined by the severity level. With UpperTolerance2 and LowerTolerance2 a second tolerance band can be defined.

NOTE: When UpperTolerance and/or LowerTolerance are used, the SetValue shall be also given.

NOTE: When UpperTolerance2 and/or LowerTolerance2 are used, UpperTolerance and/or LowerTolerance shall be also given. *UpperTolerance2* shall be between *UpperTolerance* and *MaxValue. LowerTolerance2* shall be between *LowerTolerance* and *MinValue.*

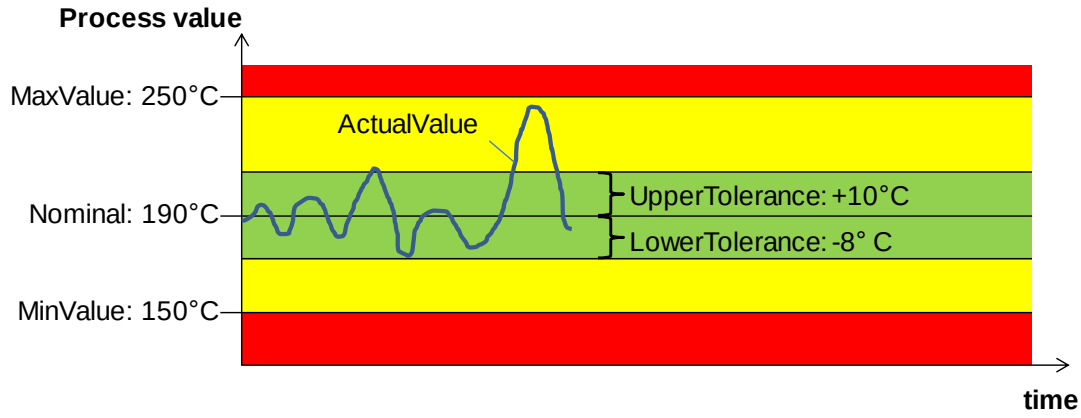


Figure 13 – Values in MonitoredParameterType (here only one tolerance band is shown)

19.6 AutomaticMonitoring

Determination if monitoring tolerance parameters are determined by auto-tuning itself (TRUE) or can be manually adjusted (FALSE). If TRUE the monitoring tolerance parameters are determined by auto-tuning regarding the set monitoring sensitivity (if used). In this case, the tolerance parameters shall then be not writeable.

19.7 MonitoringSensitivity

The monitoring sensitivity defines how closely the tolerances are set during the automatic limit setting. The *TypeDefinition* is *MultiStateValueDiscreteType*, so the *Properties EnumValues* and *ValueAsText* must be filled with the supported values out of Table 122.

Table 122 – Values for MonitoringSensitivity

EnumValue	ValueAsText	Description
0	FINE	tight tolerances
1	MIDDLE	mean tolerances
2	ROUGH	large tolerances

The absolute widths of the set tolerance bands are device dependent.

19.8 AlarmSuppression

The start-up alarm suppression deactivates alarms of a monitored parameter during start up or a setpoint jump. The *TypeDefinition* is *MultiStateValueDiscreteType*, so the *Properties EnumValues* and *ValueAsText* must be filled with the supported values out of Table 123.

Table 123 – Values for AlarmSuppression

EnumValue	ValueAsText	Description
0	OFF	no alarm suppression
1	HORN	suppressed only horn
2	COMPLETE	alarm contact, alarm via interface and horn suppressed

19.9 ResetMonitoring

Description: With this method the tolerance values are set according to the actual value and the set monitoring sensitivity. This can be used e.g. after a process change with new *SetValue* to adapt the monitoring.

Signature: `ResetMonitoring ();`

20 ControlledParameterType

The *ControlledParameterType* is used for process parameters that are controlled by the client by writing a set value and optional ramps and parameters for closed loop control.

Table 124 – ControlledParameterType Definition

Attribute	Value				
BrowseName	ControlledParameterType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *MonitoredParameterType*					
HasComponent	Variable	SetRampUp	Double	AnalogItemType	O, RW
HasComponent	Variable	SetRampDown	Double	AnalogItemType	O, RW
HasComponent	Object	ClosedLoopControl		ClosedLoopControlType	O

The variables *SetRampUp* and *SetRamp* down, which are only readable in the *MonitoredParameterType*, are writable in the *ControlledParameterType*.

If *SetValue* (already writeable in *MonitoredParameterType*) is changed by the client within the *ControlledParameterType*, the device shall control its process to reach the new value (if applicable regarding the values of *SetRampUp* and *SetRampDown*).

21 ClosedLoopControlType

With the *ClosedLoopControlType* the client can do settings for the closed loop control on the device for a parameter.

Table 125 – ClosedLoopControlType Definition

Attribute	Value				
BrowseName	ClosedLoopControlType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType*					
HasProperty	Variable	PIDParameters	PIDParametersDataType[]	PropertyType	O, RW
HasProperty	Variable	AutomaticControllerMode	Boolean	PropertyType	O, RW
HasProperty	Variable	AutoTuningActive	Boolean	PropertyType	O, R
HasComponent	Method	AutoTuningOn			O
HasComponent	Method	AutoTuningOff			O

21.1 PIDParameters

PID Parameters as array if several input signals (sensors) are used for the control.

Table 126 – PIDParametersDataType

Name	Type
PIDParametersDataType	structure
P	Double
I	Double
D	Double

21.2 AutomaticControllerMode

Determination if PID Parameters are determined by auto-tuning itself (TRUE) or can be manually adjusted (FALSE).

21.3 AutoTuningActive

This *Property* informs if the automatic tuning is currently active. TRUE during automatic tuning, FALSE when automatic tuning is finished (new PID parameters are set, if necessary). If an error occurs during automatic tuning, the PID parameters will not be changed, *AutoTuningActive* goes back to FALSE and an alarm may be sent.

21.4 AutoTuningOn

Description: Starts the self-optimisation of the controller.
Signature: `AutoTuningOn ();`

21.5 AutoTuningOff

Description: Stops an already active self-optimisation process (no control parameters are changed)
Signature: `AutoTuningOff ();`

22 MaintenanceType

The *MaintenanceType* provides information on the maintenance status. It can be used as child element of objects which represent the whole machine/device or single components, depending for which part the information is given.

Table 127 – MaintenanceType Definition

Attribute	**Value**				
BrowseName	MaintenanceType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType*					
HasProperty	Variable	Status	MaintenanceStatusEnumeration	PropertyType	M, R
HasProperty	Variable	AdditionalInformation	String	PropertyType	O, R
HasComponent	Variable	Interval	Double	AnalogItemType	O, R
HasComponent	Variable	RemainingInterval	Double	AnalogItemType	O, R
HasComponent	Method	Reset			M
HasComponent	Variable	TotalOperation	Double	AnalogItemType	O, R

22.1 Status

Maintenance status of the machine/device/component (represented by the parent element).

Table 128 – MaintenanceStatusEnumeration Definition

Value	**Description**
NOT_DUE_0	Maintenance of the device/component is not due
WARNING_1	Maintenance of the device/component is due in the near future
DUE_2	Maintenance of the device/component is due

22.2 AdditionalInformation

Additional information on the necessary maintenance. Can be also a link to another document.

22.3 Interval

Description: Regular interval between two maintenances. Use of *AnalogItemType* for flexible units (e.g. months, days, operating hours, m).
Example: 1000 h

22.4 RemainingInterval

Description: Interval before next maintenance is due.
Example: 300 h

22.5 TotalOperation

Same unit as Interval. How long is the component running in total.

22.6 Reset

This Method sets the *CurrentInterval* to 0 and *Status* to NOT_DUE_0.
Signature: `Reset ();`

23 ActiveErrorDataType

The *ActiveErrorDataType* is used for providing information about an active error in a device. It is used for minimal error handling for devices without alarm support.

Table 129 – ActiveErrorDataType Definition

Name	Type	Description
ActiveErrorDataType	structure	
Id	String	Unique identifier defined by manufacturer
Severity	UInt16	Severity as defined in the *BaseEventType* (1 = low – 1000 = high)
Message	LocalizedText	Message giving information about the error

24 HelpOffNormalAlarmType

The *HelpOffNormalAlarmType* can be used by devices, to inform the client about a *Condition* that is considered to be not normal. It is a subtype of the *OffNormalAlarmType* defined in OPC UA Part 9 and adds the Property *Helptext* to give some additional information to the operator.

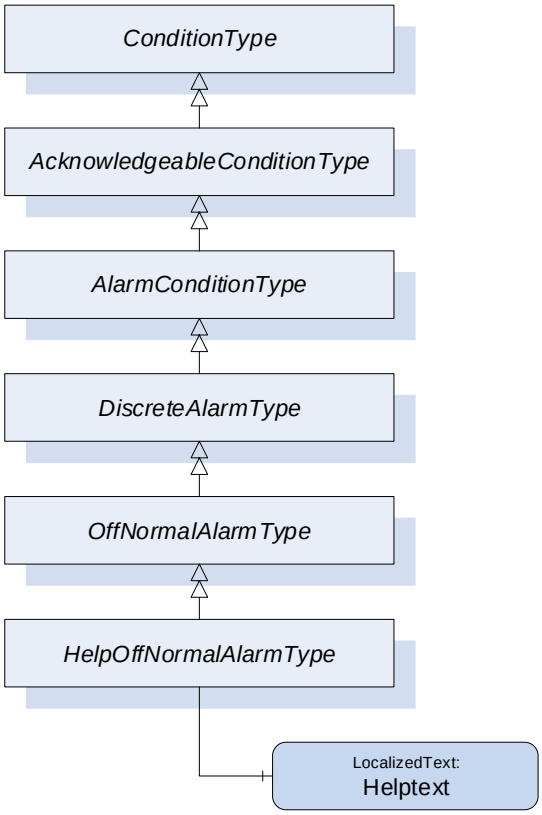


Figure 14 – HelpOffNormalAlarmType Overview

Table 130 – HelpOffNormalAlarmType Definition

Attribute	**Value**				
BrowseName	**HelpOffNormalAlarmType**				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *OffNormalAlarmType* defined in OPC UA Part 9					
HasProperty	Variable	HelpText	LocalizedText	PropertyType	M

Servers can use the *Property HelpText* also in other server specific alarms which are not derived from *OffNormalAlarmType*.