European Plastics and Rubber Machinery

EUROMAP 82.3	**OPC UA interfaces for plastics and rubber machinery – Peripheral devices – Part 3: LSR Dosing Systems**

Release Candidate 1.1, 01 November 2019

EUROMAP 82.3 (Version 1.0) is identical with
OPC 40082-3 (Edition 1.0) and VDMA 40082-1:2019-11

Contents

Figures

Tables

OPC Foundation / EUROMAP

AGREEMENT OF USE

COPYRIGHT RESTRICTIONS

- This document is provided "as is" by the OPC Foundation and EUROMAP.
- Right of use for this specification is restricted to this specification and does not grant rights of use for referred documents.
- Right of use for this specification will be granted without cost.
- This document may be distributed through computer systems, printed or copied as long as the content remains unchanged and the document is not modified.
- OPC Foundation and EUROMAP do not guarantee usability for any purpose and shall not be made liable for any case using the content of this document.
- The user of the document agrees to indemnify OPC Foundation and EUROMAP and their officers, directors and agents harmless from all demands, claims, actions, losses, damages (including damages from personal injuries), costs and expenses (including attorneys' fees) which are in any way related to activities associated with its use of content from this specification.
- The document shall not be used in conjunction with company advertising, shall not be sold or licensed to any party.
- The intellectual property and copyright is solely owned by the OPC Foundation and EUROMAP.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OPC or EUROMAP specifications may require use of an invention covered by patent rights. OPC Foundation or EUROMAP shall not be responsible for identifying patents for which a license may be required by any OPC or EUROMAP specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OPC or EUROMAP specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

WARRANTY AND LIABILITY DISCLAIMERS

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OPC FOUDATION NOR EUROMAP MAKES NO WARRANTY OF ANY KIND, EXPRESSED OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE. IN NO EVENT SHALL THE OPC FOUNDATION NOR EUROMAP BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you.

RESTRICTED RIGHTS LEGEND

This Specification is provided with Restricted Rights. Use, duplication or disclosure by the U.S. government is subject to restrictions as set forth in (a) this Agreement pursuant to DFARs 227.7202-3(a); (b) subparagraph (c)(1)(i) of the Rights in Technical Data and Computer Software clause at DFARs 252.227-7013; or (c) the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 subdivision (c)(1) and (2), as applicable. Contractor / manufacturer are the OPC Foundation, 16101 N. 82nd Street, Suite 3B, Scottsdale, AZ, 85260-1830

COMPLIANCE

The combination of EUROMAP and OPC Foundation shall at all times be the sole entities that may authorize developers, suppliers and sellers of hardware and software to use certification marks, trademarks or other special designations to indicate compliance with these materials as specified within this document. Products developed using this specification may claim compliance or conformance with this specification if and only if the software satisfactorily meets the certification requirements set by EUROMAP or the OPC Foundation. Products that do not meet these requirements may claim only that the product was based on this specification and must not claim compliance or conformance with this specification.

Trademarks

Most computer and software brand names have trademarks or registered trademarks. The individual trademarks have not been listed here.

Foreword

This specification was created by a joint working group of the OPC Foundation and EUROMAP. It is adopted identically as VDMA Specification.

NOTE: The highlighted links in Clause 2 and Annex A are not active yet and may be changed in the final version.

EUROMAP

EUROMAP is the European umbrella association of the plastics and rubber machinery industry which accounts for annual sales of around 13.5 billion euro and a 40 per cent share of worldwide production. Almost 75 per cent of its European output is shipped to worldwide destinations. With global exports of 10.0 billion euro, EUROMAP's around 1,000 machinery manufacturers are market leaders with nearly half of all machines sold being supplied by EUROMAP members.

EUROMAP provides technical recommendations for plastics and rubber machines. In addition to standards for machine descriptions, dimensions and energy measurement, interfaces between machines feature prominently. The provision of manufacturer independent interfaces ensures high levels of machine compatibility.

OPC Foundation

OPC is the interoperability standard for the secure and reliable exchange of data and information 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.

OPC UA is a platform independent service-oriented architecture that integrates all the functionality of the individual OPC Classic specifications into one extensible framework. This multi-layered approach accomplishes the original design specification goals of:

- Platform independence: from an embedded microcontroller to cloud-based infrastructure
- Secure: encryption, authentication, authorization and auditing
- Extensible: ability to add new features including transports without affecting existing applications
- Comprehensive information modelling capabilities: for defining any model from simple to complex

1 Scope

OPC 40082-3 describes the interface between injection moulding machines (IMM) and liquid silicone rubber (LSR) dosing systems for data exchange via OPC UA. The target of OPC 40082-3 is to provide a standardised interface for IMM and LSR dosing system from different manufacturers to ensure compatibility.

The following functionalities are covered:

- General information about the LSR dosing systems
- Status information
- Process data

Synchronisation of dosing between IMM and LSR dosing systems is not part of OPC 40082-3 and must be done by additional interfaces e.g. via hardwired signals.

Safety related signals like emergency stop are not included.

2 Normative references

The following referenced documents are indispensable for the application of this specification. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments and errata) applies.

OPC 10000-1: OPC Unified Architecture – Part 1: Overview and Concepts (version 1.04)

- http://www.opcfoundation.org/UA/Part1/

OPC 10000-2: OPC Unified Architecture – Part 2: Security Model (version 1.04)

- http://www.opcfoundation.org/UA/Part2/

OPC 10000-3: OPC Unified Architecture – Part 3: Address Space Model (version 1.04)

- http://www.opcfoundation.org/UA/Part3/

OPC 10000-4: OPC Unified Architecture – Part 4: Services (version 1.04)

- http://www.opcfoundation.org/UA/Part4/

OPC 10000-5: OPC Unified Architecture – Part 5: Information Model (version 1.04)

- http://www.opcfoundation.org/UA/Part5/

OPC 10000-7: OPC Unified Architecture – Part 7: Profiles (version 1.04)

- http://www.opcfoundation.org/UA/Part7/

OPC 10000-8: OPC Unified Architecture – Part 8: Data Access (version 1.04)

- http://www.opcfoundation.org/UA/Part8/

OPC 10000-9: OPC Unified Architecture – Part 9: Alarms and Conditions (version 1.04)

- http://www.opcfoundation.org/UA/Part9/

OPC 10000-11: OPC Unified Architecture – Part 11: Historical Access (version 1.04)

- http://www.opcfoundation.org/UA/Part11/

OPC 10000-100: OPC Unified Architecture – Part 100: Device Information Model (version 1.02)

- http://www.opcfoundation.org/UA/Part100/

OPC 40083: OPC UA interfaces for plastics and rubber machinery – General Type definitions (version 1.02)

- http://www.opcfoundation.org/UA/PlasticsRubber/GeneralTypes/

3 Terms, definitions and conventions

3.1 Overview

It is assumed that basic concepts of OPC UA information modelling are understood in this specification. This specification will use these concepts to describe the OPC 40082-3 Information Model. For the purposes of this document, the terms and definitions given in the documents referenced in Clause 2 apply.

NOTE: OPC UA terms and terms defined in this specification are *italicized* in the specification.

3.2 Conventions used in this document

The conventions described in OPC 40083 apply.

3.3 Abbreviations

IMM	injection moulding machine
LSR	liquid silicone rubber
LDS	LSR dosing system

4 General information to OPC UA interfaces for plastics and rubber machinery and OPC UA

For general information on OPC UA interfaces for plastics and rubber machinery and OPC UA see OPC 40083.

5 Use cases

OPC 40082-3 covers the following functionalities:

- General information about the LSR dosing system
- Status information
- Process data

6 LDS_InterfaceType

6.1 LDS_InterfaceType Definition

This OPC UA *ObjectType* is used for the root *Object* representing a LSR dosing system with its subcomponents. It is formally defined in Table 1.

NOTE: To promote interoperability of *Client*s and *Server*s, all instantiated *Devices* shall be aggregated in an *Object* called "DeviceSet" (see OPC UA for Devices)

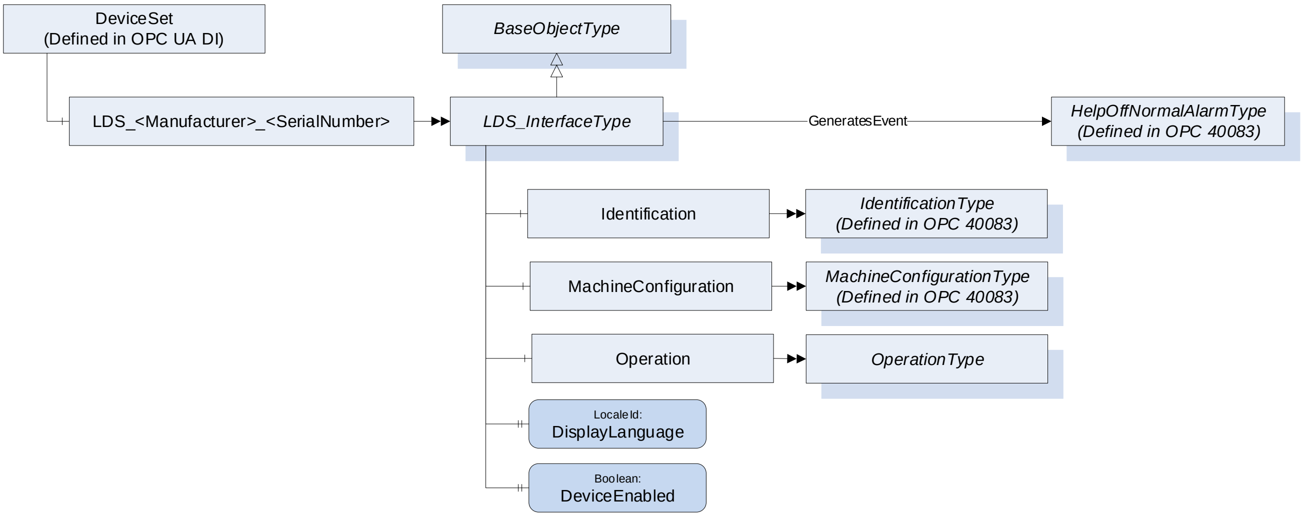

Figure 1 – LDS_InterfaceType Overview

Table 1 – LDS_InterfaceType Definition

Attribute	**Value**				
BrowseName	LDS_InterfaceType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType*					
HasComponent	Object	Identification		IdentificationType	M
HasComponent	Object	MachineConfiguration		MachineConfigurationType	M
HasComponent	Object	Operation		OperationType	M
HasProperty	Variable	DisplayLanguage	LocaleId	PropertyType	O, RW
HasProperty	Variable	DeviceEnabled	Boolean	PropertyType	O, RW
GeneratesEvent	ObjectType	HelpOffNormalAlarmType	Defined in OPC 40083		

The *BrowseName* of the object instance shall be "LDS_<Manufacturer>_<SerialNumber>"

Example: "LDS_Reinhardt_0123456".

NOTE: The namespace of this *BrowseName* is the local server URI with namespace index 1 or a vendor specific namespace with server specific namespace index (see Table 24). The *BrowseNames* of the nodes below are in the namespace of the specification where used Type is defined.

Example:

BrowseName	**Namespace**	**Namespace index**	**Remarks**
LDS_Reinhardt_0123456	Local Server URI or vendor specific namespace	1 or server specific	OPC 40082-3 only defines the *LDS_InterfaceType*. The instance is generated in the local server

↓

Identification	http://opcfoundation.org/UA/PlasticsRubber/LDS/	server specific	The object *Identification* is a child of *LDS_InterfaceType* which is defined in OPC 40082-3

↓

Manufacturer	http://opcfoundation.org/UA/PlasticsRubber/GeneralTypes/	server specific	The variable *Manufacturer* is a child of *IdentificationType* which is defined in OPC 40083.

6.2 DisplayLanguage

With the *DisplayLanguage Property* the client can set the desired language on the user interface at the LDS. If the peripheral device does not support the configured language, it can keep the previous setting or use English as the default.

6.3 DeviceEnabled

The variable *DeviceEnabled* is used to release the drives of the dosing system. If the value is FALSE, the LDS shall not be able to start ist drives.

7 Identification

The *IdentificationType* for the identification of the device is defined in OPC 40083. All mandatory nodes shall be filled with valid values from the server.

The *DeviceClass Property* in the *Identification Object* shall have the value " LSR Dosing System"

8 MachineConfiguration

The *MachineConfiguration Object* represents the current configuration of the LDS.
The *MachineConfigurationType* is defined in OPC 40083.

9 OperationType

This *ObjectType* contains components which are necessary to operate the LDS. It is formally defined in Table 2.

Table 2 – OperationType Definition

Attribute	**Value**				
BrowseName	OperationType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseObjectType*					
HasProperty	Variable	DeviceMappingNumber	UInt32	PropertyType	M, RW
HasComponent	Method	IdentifyDevice			O
HasProperty	Variable	HighestActiveAlarmSeverity	UInt16	PropertyType	M, R
HasComponent	Variable	ActiveErrors	ActiveErrorDataType[]	BaseDataVariableType	M, R
HasComponent	Method	ResetAllErrors			O
HasComponent	Method	ResetErrorById			O
HasProperty	Variable	NumberOfAdditives	UInt32	PropertyType	M, R
HasProperty		MaterialBalanceSystemType	MaterialBalanceSystemTypeEnumeration	PropertyType	M, R
HasProperty	Variable	ActivateMaterialBalanceSystem	Boolean	PropertyType	O, RW
HasComponent	Variable	DeliveryType	UInt16	MultiStateValueDiscreteType	M, RW
HasComponent	Object	DeliveryPressure		ControlledParameterType	O
HasComponent	Variable	DeliveryPressureMeasuringPoint	UInt16	MultiStateValueDiscreteType	O, RW
HasComponent	Object	DeliveryFlowrate		ControlledParameterType	O
HasComponent	Variable	ActualShotWeight	Double	AnalogItemType	O, R
HasComponent	Variable	SetShotWeight	Double	AnalogItemType	O, RW
HasComponent	Variable	SetValueCompositeDensity	Double	AnalogItemType	O, RW
HasComponent	Variable	MaxDeviationMixingRatio	Double	AnalogItemType	O, RW
HasComponent	Variable	TargetDeviationMixingRatio	Double	AnalogItemType	O, R
HasComponent	Variable	ActualDeviationMixingRatio	Double	AnalogItemType	O, R
HasComponent	Variable	RemainingMaterialTime	Duration	BaseDataVariableType	O, R
HasComponent	Variable	PurgeMode	UInt16	MultiStateValueDiscreteType	O, RW
HasProperty	Variable	PurgeStatus	PurgeStatusEnumeration	PropertyType	O, R
HasProperty	Variable	ActivateRemoteControl	Boolean	PropertyType	M, RW
HasProperty	Variable	RemoteControlActivated	Boolean	PropertyType	M, R
HasComponent	Object	Component_<X>		ComponentType	MP
HasComponent	Object	Additive_<Y>		AdditiveType	OP
GeneratesEvent	ObjectType	LDSCycleParametersEventType	Defined in 9.22		

The *BrowseName* of *ComponentType* shal be built of “Component_” and a character ‘A’, ‘B’ , … (e.g. Component_A, Component_B).

The *BrowseName* of *AdditiveType* shall be built of “Additive_” and a number from 1 to n. (e.g. Additive_1).

9.1 DeviceMappingNumber

Description: Unique identifier/address/number for devices of the same *DeviceType* within a local network. Several peripheral devices of the same *DeviceType* can be connected to an IMM. In most applications, the IMM must map the connected peripheral devices to internal logical devices and zones in a fixed configuration (e.g. hot runner systems according to the wiring or temperature control devices according to the tubing).

The mapping shall be stable after reconnecting the devices and is therefore not possible via IP addresses, which can be assigned dynamically via DHCP. *DeviceMappingNumber* sets the mapping order of peripheral devices of the same type on the local network and is therefore of type *UInt32*.

Example: 1

9.2 IdentifyDevice

Description: The peripheral device on which this method is called shows itself by e.g. activation of a LED.

Signature:

```
IdentifyDevice ();
```

9.3 HighestActiveAlarmSeverity

Description: Indication of the severity of the highest active alarm (0 = no active alarm – 1000 = possible error). It provides a minimal error handling for devices without alarm support. However, the variable shall be filled even if alarms are supported.

Example: 400

9.4 ActiveErrors

Description: List of the active errors of the device. It provides a minimal error handling for devices without alarm support. However, the variable shall be filled even if alarms are supported. The *ActiveErrorDataType* is defined in OPC 40083. If there is no active error, the array is empty.

9.5 ResetAllErrors

Description: Method to reset all errors of the device.

Signature:

```
ResetAllErrors();
```

9.6 ResetErrorById

Description: Method to reset one error of the device.

Signature:

```
ResetErrorById(
       [in]   String Id);
```

Table 3 –ResetErrorById Method Arguments

Argument	Description
Id	Id of the error, listed in *ActiveErrors*, that shall be reset.

Table 4 – ResetErrorById Method AddressSpace Definition

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

9.7 NumberOfAdditives

Number of the physically present additive dosing systems.

Static number, given by the device.

9.8 MaterialBalanceSystemType

Table 5 – MaterialBalanceSystemTypeEnumeration

Value	Description
NOT_AVAILABLE_0	No material balance system available on the LDS. *ActivateMaterialBalanceSystem* is not present, because it is not possible to switch a material balance system on
ALWAYS_ACTIVE_1	Material balance system is available on the LDS and always active. *ActivateMaterialBalanceSystem* is not present, because it is not possible to switch a material balance system off
SELECTABLE_2	Material balance system is available on the LDS and it can be switched on and off via the interface via the present *ActivateMaterialBalanceSystem*.

9.9 ActivateMaterialBalanceSystem

If the value is true, the material balance system is activated.

9.10 DeliveryType

The dosing system works with delivery pressure or volumetric flow. As some LSR dosing systems support the selection of the *DeliveryType*, the *Property* can be writeable. Therefore, the *TypeDefinition* is *MultiStateValueDiscreteType*, so the *Properties EnumValues* and *ValueAsText* must be filled with the supported values out of Table 6.

Table 6 – Values for DeliveryType

EnumValue	ValueAsText	Description
0	PRESSURE	Dosing system with delivery pressure
1	VOLUMETRIC_FLOWRATE	Dosing system with volumetric flow

9.11 DeliveryPressure, DeliveryPressureMeasuringPoint

With the object *DeliveryPressure* and *DeliverPressureMeasuringPoint* the client can set (and monitor) the delivery pressure of the LDS. Both are optional, but the two elements shall always be used together.

For systems with *DeliveryPressure* the components *SetValue*, *UpperTolerance* and *LowerTolerance* defined in the *ControlledParameterType* are mandatory. If the upper or lower tolerance band is passed it is documented in the *ErrorStatus*.

Unit: bar or psi (=lbf/in^2)

The variable *DeliveryPressureMeasuringPoint* represents the position of the pressure sensor. As some LSR dosing systems support the selection of the position, the *Property* can be writeable. Therefore, the *TypeDefinition* is *MultiStateValueDiscreteType*, so the *Properties EnumValues* and *ValueAsText* must be filled with the supported values out of Table 7.

Table 7 – Values for PressureMeasuringPoint

EnumValue	ValueAsText	Description
0	PUMP_A	Pressure sensor position pump A
1	PUMP_B	Pressure sensor position pump B
2	BLENDER	Pressure sensor position blender
3	MANUAL	Pressure is manually adjusted

9.12 DeliveryFlowrate

For system with delivery volumetric flow rate the components *SetValue*, *UpperTolerance* and *LowerTolerance* are mandatory. If the upper or lower tolerance band is passed it is documented in the *ErrorStatus*.

Unit: l/h or gal/h

9.13 ActualShotWeight

Specifies the value determined by the feeder as the shot weight.

Unit: g or lb

9.14 SetShotWeight

Reference value determined by the IMM or defined by the user on the IMM side.

Unit: g or lb

9.15 SetValueCompositeDensity

The composite set point of density.

Unit: g/cm³ or lb/in³

9.16 MaxDeviationMixingRatio, TargetDeviationMixingRatio, ActualDeviationMixingRatio

If a material balance system is used these variables are used to set and monitor the deviation from the set mixing ration of component A and B.

NOTE: The mixing ration itself (which is usually 50% component A and 50% component B) is set in the *ComponentType* (see 10, variable *PercentageComponent* and method *SetPercentageComponent*).

MaxDeviationMixingRatio is writeable by the client and used to limit the maximum deviation in percent.

TargetDeviationMixingRatio: This deviation (in percent) is set/used by the material balance system

ActualDeviationMixingRatio: Actual deviation (in percent)

9.17 RemainingMaterialTime

Remaining time until first material is empty.

9.18 PurgeMode

Depending on this preselected PurgeMode, various purge function can be activated via the dosing signal of the IMM.

Table 8 – Values for PurgeMode

EnumValue	ValueAsText	Description
0	OFF	No purge function. Normal dosing via dosing signal.
1	WITH_COMPONENT_A	Purge A
2	WITH_COMPONENT_B	Purge B
3	WITH_COMPONENT_A_B	Venting
4	WITH_COMPONENT_A_OR_B	System choose the component which is used for purging (usually the component with the larger remaining quantity)
5	CYCLIC_COMPONENT_A_B	Purge A and B cyclic

9.19 PurgeStatus

Actual status of the purge function. *PurgeStatus* must show *OFF* if no purge function is active.

Purge functions can be activated by the operator or via dosing signal of the IMM depending on the *PurgeMode*.

Table 9 - PurgeStatusEnumeration

Value	Description
OFF_0	No purge function is active.
COMPONENT_A_1	Purge component A is active.
COMPONENT_B_2	Purge component B is active.
COMPONENT_A_AND_B_3	Venting
COMPONENT_A_AND_B_CYCLIC_4	Cyclic purge component A and B is active.

9.20 ActivateRemoteControl

Client sets ActivateRemoteControl *true* to control the device with the signal interface (I/O) from the injection moulding machine (“automatic mode”).

NOTE: Synchronisation of dosing between IMM and LSR dosing systems is not part of OPC 40082-3 and must be done by additional interfaces e.g. via hardwired signals.

9.21 RemoteControlActivated

RemoteControlActivated is *true* if the LSR dosing system is ready to be controlled via signal interface (I/O).

9.22 LDSCycleParametersEventType

The *LDSCycleParametersEventType* represents information on a dosing cycle. It is fired after each finished dosing cycle of the LDS.

The *LDSCycleParametersEventType* is formally defined in Table 10.

Table 10 – LDSCycleParametersEventType Definition

Attribute	**Value**				
BrowseName	LDSCycleParametersEventType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *BaseEventType* defined in OPC UA Part 5					
HasProperty	Variable	CycleNumber	UInt64	PropertyType	M
HasProperty	Variable	DosingTime	Duration	PropertyType	O
HasComponent	Variable	MixingRatioTarget	Double	AnalogItemType	O
HasComponent	Variable	MixingRatioActual	Double	AnalogItemType	O
HasComponent	Variable	AdditivesRatioTarget	Double[]	AnalogItemType	O
HasComponent	Variable	MixingRatioBalance	Double	AnalogItemType	O
HasComponent	Variable	AdditivesRatioActual	Double[]	AnalogItemType	O
HasComponent	Variable	VolumeA	Double	AnalogItemType	O
HasComponent	Variable	VolumeB	Double	AnalogItemType	O
HasComponent	Variable	VolumeAB	Double	AnalogItemType	O
HasComponent	Variable	VolumeAdditives	Double[]	AnalogItemType	O
HasComponent	Variable	VolumeTotal	Double	AnalogItemType	O
HasComponent	Variable	ResidualAmountA	Double	AnalogItemType	O
HasComponent	Variable	ResidualAmountB	Double	AnalogItemType	O
HasComponent	Variable	MixingPointPressureA	Double	AnalogItemType	O
HasComponent	Variable	MixingPointPressureB	Double	AnalogItemType	O
HasComponent	Variable	MixingPointPressureBlender	Double	AnalogItemType	O
HasComponent	Variable	AdditivesPressure	Double[]	AnalogItemType	O
HasComponent	Variable	FilterPressurePrimary	Double	AnalogItemType	O
HasComponent	Variable	FilterPressureSecondary	Double	AnalogItemType	O

9.22.1 Cycle Number

Number of the dosing cycle. Gets counted up after each dosing cycle.

Example: 900

9.22.2 DosingTime

Duration of the dosing cycle.

9.22.3 MixingRatioTarget

Target mixing ratio of the last cycle. (includes ratio change when MaterialBalanceSystem is active. The ration is calculated: A/B

Examples: 1 (A 50 : 50 B) → without MaterialBalanceSystem
1,05 (A 51,25 : 48,75 B) → active MaterialBalanceSystem

9.22.4 MixingRatioActual

Actual mixing Ratio from A&B component. The ration is calculated: A/B

Example: 1,02.

9.22.5 MixingRatioBalance

Actual Mixing Ration Shift cause of Material Balance System.

Example: MixingRatioTarget = 1, MixingRatioActual = 1,05 → MixingRatioBalance = 0,05.

9.22.6 AddivtivesRatioTarget

Target ratios of additives in percentage which are set in AdditiveFraction of AdditiveType

9.22.7 AdditivesRatioActual

Actual ratios of additive in percentage.

Example: [2,1 % ; 1,2 %]

9.22.8 VolumeA

Volume of component A that was added to the process in the last cycle.

Unit: cm³ or in³

9.22.9 VolumeB

Volume of component B that was added to the process in the last cycle.

Unit: cm³ or in³

9.22.10 VolumeAB

Volume of component A&B that was added to the process in the last cycle

Unit: cm³ or in³

9.22.11 VolumeAdditives

Volumes of the additives that were added to the process in the last cycle

Unit: cm³ or in³

9.22.12 VolumeTotal

Volume of all components (A+B+ all additives)

Unit: cm³ or in³

9.22.13 ResidualAmountA

Residual weight amount of component A at the end of the dosing cycle.

Unit: kg or lb

9.22.14 ResidualAmountB

Residual weight amount of component B at the end of the dosing cycle.

Unit: kg or lb

9.22.15 MixingPointPressureA

Average pressure of component A during the last cycle at the blender,

Unit: bar or psi

9.22.16 MixingPointPressureB

Average pressure of component B during the last cycle at the blender

Unit: bar or psi

9.22.17 MixingPointPressureBlender

Average pressure of component A&B during the last cycle at the blender.

Unit: bar or psi

9.22.18 AdditivesPressure

Average pressure of the additive during the last cycle at the measuring point.

Unit: bar or psi

9.22.19 FilterPressurePrimary, FilterPressureSecondary

Average material pressure during the last cycle before and after the filter. The Pressure difference between FilterPressurePrimary & FilterPressureSecondary can be used to check if the filter is blocked/ will be blocked soon/ has to be maintained. Unit: bar or psi

10 ComponentType

Table 11 - ComponentType Definition

Attribute	**Value**				
BrowseName	ComponentType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rules**
Subtype of *BaseObjectType*					
HasComponent	Variable	SetValueDensity	Double	AnalogItemType	O, R
HasComponent	Method	SetSetValueDensity			O
HasComponent	Variable	PercentageComponent	Double	AnalogItemType	O, R
HasComponent	Method	SetPercentageComponent			O
HasComponent	Variable	ActualPressure	Double	AnalogItemType	O, R
HasComponent	Variable	ResidualAmount	Double	AnalogItemType	O, R
HasComponent	Variable	RemainingMaterialTime	Duration	BaseDataVariable Type	O, R
HasProperty	Variable	AllowsCycles	Double	PropertyType	O, R
HasProperty	Variable	Status	ComponentStatusEnumeration	PropertyType	M, R

10.1 SetValueDensity

Set point material density.

Unit: g/cm³ or lb/in³

10.2 SetSetValueDensity

This optional method is used to modify *SetValueDensity* if allowed by the device.

Signature:

```
SetValueDensity (
   [in]    Double      Density);
```

Table 12 – SetValueDensity Method Arguments

Argument	**Description**
Density	New set point of the material density. Note: The *DataType* is Double. The unit is specified in the Variable *SetValueDensity.*

Table 13 – SetValueDensity Method AddressSpace Definition

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

10.3 PercentageComponent

Percentage of the component from *0* to *100*. In standard application with the same proportion of component A and B the value is 50.0 for component A and 50.0 for component B.

10.4 SetPercentageComponent

This optional method is used to modify *PercentageComponent* if allowed by the device.

Signature:

```
SetPercentageComponent (
   [in]    Double     Percentage);
```

Table 14 – SetPercentageComponent Method Arguments

Argument	**Description**
Percentage	New percentage of the material from 0 to 100.

Table 15 – SetPercentageComponent Method AddressSpace Definition

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

10.5 ActualPressure

Actual pressure of the component.

Unit: bar or psi

10.6 ResidualAmount

Residual amount of the material.

Unit: kg or lb

10.7 RemainingMaterialTime

Time until the material of the component is empty.

10.8 AllowsCycles

Expected number of remaining cycles with the current drum.

10.9 Status

Actual status of the component provides a minimal error handling for devices without event support.

Detailed information may be published via *ComponentOffNormalAlarmType*.

Table 16 - ComponentStatusEnumeration

Value	Description
GOOD_0	Component has no error or warning.
WARNING_1	The component has an undefined warning, but no need to stop the production. Detailed information may be published via an alarm (*HelpOffNormalAlarmType)*.
WARNING_PRESSURE_TOO_HIGH_2	Pressure is too high. No need to stop the process but influence to the part quality.
WARNING_PRESSURE_TOO_LOW_3	Pressure is too low. No need to stop the process but influence to the part quality.
ADVANCE_WARNING_DRUM_CHANGE_4	Warning, barrel change is imminent. No need to stop the process.
ERROR_DRUM_EMPTY_5	Drum of the component is empty. Production need to be stopped.
ERROR_6	The component has an error and process needs to be stopped. Detailed information may be published via an alarm (*HelpOffNormalAlarmType*).

11 AdditiveType

Table 17 - AdditiveType Definition

Attribute	Value				
BrowseName	AdditiveType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rules**
Subtype of *BaseObjectType*					
HasProperty	Variable	ActivateAdditive	Boolean	PropertyType	M, RW
HasProperty	Variable	AdditiveActivated	Boolean	PropertyType	M, R
HasProperty	Variable	Status	AdditiveStatusEnumeration	PropertyType	M, R
HasComponent	Object	AdditiveFraction		ControlledParameterType	O
HasComponent	Object	AdditiveVolume		ControlledParameterType	O

11.1 ActivateAdditive

SetValue to activate the additive.

11.2 AdditiveActivated

Is *true* if the additive is activated.

11.3 Status

Actual status of the additive provides a minimal error handling for devices without event support.

Detailed information may be published via *AdditiveOffNormalAlarmType*.

Table 18 - AdditiveStatusEnumeration

Value	Description
GOOD_0	Additive has no error or warning
WARNING_1	The additive has an undefined warning, but no need to stop the production. Detailed information may be published via an alarm (*HelpOffNormalAlarmType)*.
ADVANCE_WARNING_ADDITIVE_CHANGE_2	Warning, additive change is imminent. No need to stop the process.
ERROR_EMPTY_3	Error, the additive is empty. Production need to be stopped.
ERROR_4	The additive has an error and process needs to be stopped. Detailed information may be published via an alarm (*HelpOffNormalAlarmType*).

11.4 AdditiveFraction

Contains the SetValue, ActualValue, LowerTolerance and UpperTolerance of the additive fraction in percent.

11.5 AdditiveVolume

Defines the value of additive per shot/stroke. Total amount stays the same (defined by *AdditiveFraction*). Used to distribute the total amount to several strokes.

Unit: cm^3 or in^3

12 Alarmmanagement

As defined in OPC 40083, the root node of the specific interface, e.g. an instance of *LDS_InterfaceType*, set the *SubscribeToEvents* flag in the *EventNotifier* attribute.

The client subscribes to events at this root node and receives the events already defined in this specification, such as temperature limit alarms or diagnostic events.

A LDS may optionally generate additional manufacturer-specific alarms, warnings or information displayed on the user interface of the device and can publish these events via two special *AlarmTypes*.

Component-related messages should be represented by instances of *ComponentOffNormalAlarmType*, additive-related messages should be represented by instances of *AdditiveOffNormalAlarmType*, other device information is of type *HelpOffNormalAlarmType*.

All are subtypes of *OffNormalAlarmType,* can be synchronized via *ConditionRefresh* and contain a *Severity* for error handling according to OPC 40083.

12.1 ComponentOffNormalAlarmType

The *ComponentOffNormalAlarmType* represent component-related text messages (alarms, error messages, warnings, information) of the peripheral device and is a subtype of *HelpOffNormalAlarmType* as defined in OPC 40083.

NOTE: For messages related to the whole device, the *HelpOffNormalAlarmType* shall be used.

Table 19 – ComponentOffNormalAlarmType Definition

Attribute	**Value**				
BrowseName	ComponentOffNormalAlarmType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *HelpOffNormalAlarmType* defined in OPC 40083					
HasProperty	Variable	ComponentId	NodeId	PropertyType	M, R

ComponentId specifies the *NodeId* of the related Component. In case of medium or high severity, the IMM can sort out bad parts or stop production.

12.2 AdditiveOffNormalAlarmType

The *AdditiveOffNormalAlarmType* represent additive-related text messages (alarms, error messages, warnings, information) of the peripheral device and is a subtype of *HelpOffNormalAlarmType*.

Table 20 – AdditiveOffNormalAlarmType Definition

Attribute	**Value**				
BrowseName	AdditiveOffNormalAlarmType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of *HelpOffNormalAlarmType* defined in OPC 40083					
HasProperty	Variable	AdditiveId	NodeId	PropertyType	M, R

AdditiveId specifies the *NodeId* of the related additive. In case of medium or high severity, the IMM can sort out bad parts or stop production.

13 Profiles and Namespaces

13.1 Namespace Metadata

Table 21 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 Part5 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 Part5.

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

Table 21 – NamespaceMetadata Object for this Specification

Attribute	**Value**		
BrowseName	http://opcfoundation.org/UA/PlasticsRubber/LDS/		
References	**BrowseName**	**DataType**	**Value**
HasProperty	NamespaceUri	String	http://opcfoundation.org/UA/PlasticsRubber/LDS/
HasProperty	NamespaceVersion	String	RC 1.00.1
HasProperty	NamespacePublicationDate	DateTime	2019-09-10 12:00:00
HasProperty	IsNamespaceSubset	Boolean	False
HasProperty	StaticNodeIdTypes	IdType[]	{Numeric}
HasProperty	StaticNumericNodeIdRange	NumericRange[]	Null
HasProperty	StaticStringNodeIdPattern	String	Null

13.2 Conformance Units and Profiles

This chapter defines the corresponding profiles and conformance units for the OPC UA Information Model for OPC 40082-3. *Profiles* are named groupings of conformance units. Facets are profiles that will be combined with other *Profiles* to define the complete functionality of an OPC UA *Server* or *Client.* The following tables specify the facets available for *Servers* that implement the OPC 40082-3 Information Model companion specification.

NOTE: The names of the supported profiles are available in the *Server Object* under *ServerCapabilities.ServerProfileArray*

Table 22 – *OPC 40082-3 Basic Server Facet* Definition

Conformance Unit	**Description**	**Optional/ Mandatory**
OPC 40082-3 Basic	Support of *LDS_InterfaceType* and all mandatory child elements giving information on the LSR dosing system itself, the current configuration and status.	M
Profile		
ComplexType Server Facet (defined in OPC UA Part 7)		M
Method Server Facet (defined in OPC UA Part 7)		M
BaseDevice_Server_Facet (defined in OPC UA Part 100)		M

Table 23 – *OPC 40082-3 Alarms Server Facet* Definition

Conformance Unit	**Description**	**Optional/ Mandatory**
OPC 40082-3 Alarms	Support of *HelpOffNormalAlarmType* and *ComponentZoneOffNormalAlarmType* providing error information. If this facet is supported and a client subscribes to the events, the server shall provide all errors via alarms in addition to the error variables included in the *OperationType*	M
A & C Alarm Server Facet (defined in OPC UA Part 7)		M

13.3 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 *AddressSpace* 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 24 provides a list of mandatory and optional namespaces used in an OPC 40082-3 OPC UA *Server*.

Table 24 – Namespaces used in an OPC 40082-3 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://opcfoundation.org/UA/PlasticsRubber/GeneralTypes/	Namespace for *NodeIds* and *BrowseNames* defined in OPC 40083. The namespace index is server specific.	Mandatory
http://opcfoundation.org/UA/PlasticsRubber/LDS/	Namespace for *NodeIds* and *BrowseNames* defined in this specification. The namespace index is server specific.	Mandatory
Vendor specific types and instances	A server may provide vendor specific types like types derived from *MachineType* or *MachineStatusType* or vendor specific instances of devices in a vendor specific namespace.	Optional

Table 25 provides a list of namespaces and their index used for *BrowseNames* in this specification. The default namespace of this specification is not listed since all *BrowseNames* without prefix use this default namespace.

Table 25 – Namespaces used in this specification

NamespaceURI	Namespace Index	Example
http://opcfoundation.org/UA/	0	0:NodeVersion

Annex A
(normative)

OPC 40082-3 Namespace and mappings

A.1 Namespace and identifiers for OPC 40082-3 Information Model

This appendix defines the numeric identifiers for all of 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://opcfoundation.org/UA/PlasticsRubber/LDS/

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

- http://www.opcfoundation.org/UA/schemas/PlasticsRubber/LDS/1.0/NodeIds.csv

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

- http://www.opcfoundation.org/UA/schemas/PlasticsRubber/LDS/NodeIds.csv

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

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

- http://www.opcfoundation.org/UA/schemas/PlasticsRubber/LDS/1.0/Opc.Ua.PlasticsRubber.LDS.NodeSet2.xml

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

- http://www.opcfoundation.org/UA/schemas/PlasticsRubber/LDS/Opc.Ua.PlasticsRubber.LDS.NodeSet2.xml