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 1.01.0, 2022-07-22

EUROMAP 82.3 (Release Candidate 1.01.0) is identical with

OPC 40082-3 (Release Candidate 1.01.0) and VDMA 40082-3:2022-09

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

Contents

Figures

Tables

OPC Foundation / EUROMAP

AGREEMENT OF USE

TRADEMARKS

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

GENERAL PROVISIONS

Should any provision of this Agreement be held to be void, invalid, unenforceable or illegal by a court, the validity and enforceability of the other provisions shall not be affected thereby.

This Agreement shall be governed by and construed under the laws of Germany.

This Agreement embodies the entire understanding between the parties with respect to, and supersedes any prior understanding or agreement (oral or written) relating to, this specification.

Foreword

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

Compared with the previous version, the following changes have been made:

Version	Changes
OPC 40082-3, version 1.00.1 (identical with VDMA 40083:2021-09 and EUROMAP 82-3, version 1.00.1)	NodeSet corrected (falsely used ModellingRule mandatory for Variables ActivateClosedLoopControl and ClosedLoopControlActivated in ObjectType AdditiveType) → corrected to be aligned with document → new NamespaceVersion and NamespacePublicationDate (see Mantis ID 7037)
OPC 40082-3, version RC 1.01.0 (identical with Draft VDMA 40083:2022-09 and EUROMAP 82-3, version RC 1.01.0)	In OperationType optional *Variable DosingActive* added In *ComponentType* – optional *Variables ActualFollowerPlatePressure*, *ActualFollowerPlatePressure* and *DrumCapacity* added – description of *ActualPressure* clarified

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 documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies

OPC 10000-3, *OPC Unified Architecture - Part 3: Address Space Model*

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

OPC 10000-5, *OPC Unified Architecture - Part 5: Information Model*

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

OPC 10000-6, *OPC Unified Architecture - Part 6: Mappings*

http://www.opcfoundation.org/UA/Part6/

OPC 10000-7, *OPC Unified Architecture - Part 7: Profiles*

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

OPC 10000-8, *OPC Unified Architecture - Part 8: Data Access*

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

OPC 10000-9, *OPC Unified Architecture - Part 9: Alarms and Conditions*

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

OPC 10000-100, *OPC Unified Architecture - Part 100: Devices*

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

OPC 40083: OPC UA interfaces for plastics and rubber machinery – General Type definitions

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 that 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)

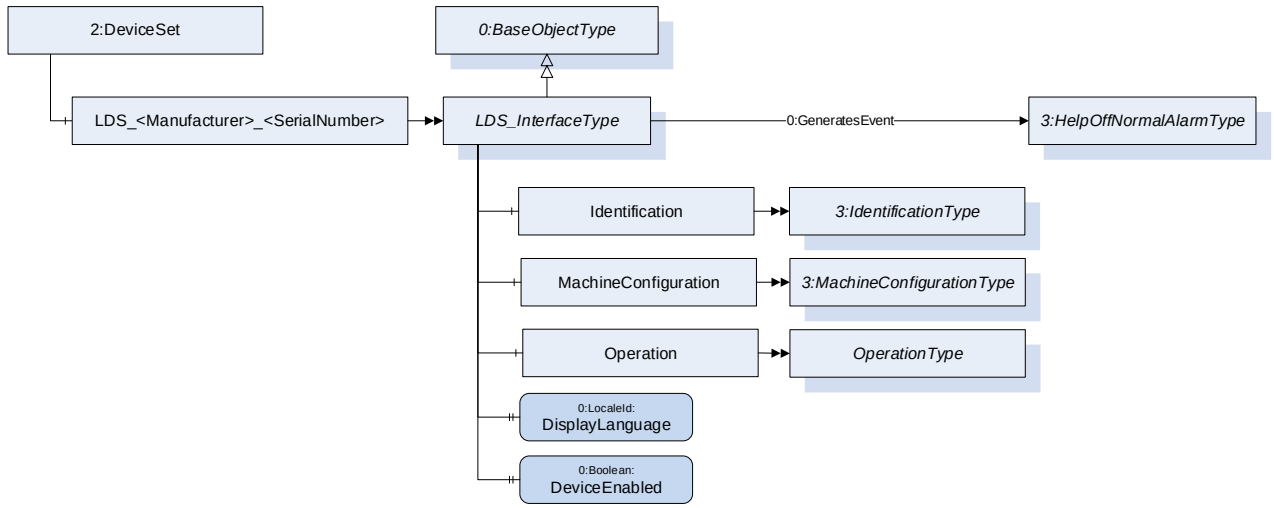

Figure 1 – LDS_InterfaceType Overview

Table 1 – LDS_InterfaceType Definition

Attribute	Value				
BrowseName	LDS_InterfaceType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:*BaseObjectType* defined in OPC UA Part 5					
0:HasComponent	Object	Identification		3:IdentificationType	M
0:HasComponent	Object	MachineConfiguration		3:MachineConfigurationType	M
0:HasComponent	Object	Operation		OperationType	M
0:HasProperty	Variable	DisplayLanguage	0:LocaleId	0:PropertyType	O, RW
0:HasProperty	Variable	DeviceEnabled	0:Boolean	0:PropertyType	O, RW
0:GeneratesEvent	ObjectType	0: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 30). 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/DI/	server specific	The variable *Manufacturer* is a child of *IdentificationType* which is defined in OPC 40083. However, it derives from the *ComponentType* defined in OPC 10000-100. The Variable *Manufacturer* is defined there.

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**	**Other**
Subtype of 0:*BaseObjectType* defined in OPC UA Part 5					
0:HasProperty	Variable	DeviceMappingNumber	0:UInt32	0:PropertyType	M, RW
0:HasComponent	Method	IdentifyDevice			O
0:HasProperty	Variable	HighestActiveAlarmSeverity	0:UInt16	0:PropertyType	M, RO
0:HasComponent	Variable	ActiveErrors	3:Classified ActiveError DataType[]	0:BaseDataVariableType	M, RO
0:HasComponent	Method	ResetAllErrors			O
0:HasComponent	Method	ResetErrorById			O
0:HasComponent	Method	SetCycleNumber			O
0:HasProperty	Variable	MaterialBalanceSystemType	MaterialBalance SystemType Enumeration	0:PropertyType	M, RO
0:HasProperty	Variable	ActivateMaterialBalance System	0:Boolean	0:PropertyType	O, RW
0:HasComponent	Variable	DeliveryType	0:UInt16	0:MultiStateValue DiscreteType	M, RW
0:HasComponent	Object	DeliveryPressure		3:ControlledParameterType	O
0:HasComponent	Variable	DeliveryPressure MeasuringPoint	0:UInt16	0:MultiStateValueDiscreteType	O, RW
0:HasComponent	Object	DeliveryFlowrate		3:ControlledParameterType	O
0:HasComponent	Variable	ActualShotWeight	0:Double	0:AnalogItemType	O, RO
0:HasComponent	Variable	SetShotWeight	0:Double	0:AnalogItemType	O, RW
0:HasComponent	Variable	SetValueCompositeDensity	0:Double	0:AnalogItemType	O, RW
0:HasComponent	Variable	MixingRatioTarget	0:Double	0:AnalogItemType	O, RW
0:HasComponent	Variable	MaxDeviationMixingRatio	0:Double	0:AnalogItemType	O, RW
0:HasComponent	Variable	TargetDeviationMixingRatio	0:Double	0:AnalogItemType	O, RO
0:HasComponent	Variable	ActualDeviationMixingRatio	0:Double	0:AnalogItemType	O, RO
0:HasComponent	Variable	RemainingMaterialTime	0:Duration	0:BaseDataVariableType	O, RO
0:HasComponent	Variable	PurgeMode	0:UInt16	0:MultiStateValue DiscreteType	O, RW
0:HasProperty	Variable	PurgeStatus	PurgeStatus Enumeration	0:PropertyType	O, RO
0:HasComponent	Variable	ActivateRemoteControl	0:UInt16	0:MultiStateValueDiscreteType	M, RW
0:HasComponent	Variable	RemoteControlActivated	0:UInt16	0:MultiStateValueDiscreteType	M, RO
0:HasComponent	Method	StartDosing			O
0:HasComponent	Method	StopDosing			O
0:HasComponent	Variable	DosingActive	0:Boolean	0:BaseDataVariableType	O, RO
0:HasComponent	Object	Component_A		ComponentType	M
0:HasComponent	Object	Component_B		ComponentType	M
0:HasComponent	Object	Additive_<Y>		AdditiveType	OP
0:GeneratesEvent	Object Type	LDSCycleParameters EventType	Defined in 9.24		

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 ();
```

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

Table 3 – IdentifyDevice Method AddressSpace Definition

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

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 *ClassifiedActiveErrorDataType* 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();
```

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

Table 4 – ResetAllErrors Method AddressSpace Definition

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

9.6 ResetErrorById

Description: Method to reset one error of the device.

Signature:

```
ResetErrorById(
   [in]    0:String          Id);
```

Table 5 – ResetErrorById Method Arguments

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

Table 6 – ResetErrorById Method AddressSpace Definition

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

9.7 SetCycleNumber

Description: Method to set the cycle number of the LDS to synchronize it with the cycle number of the injection moulding machine.

Signature:

```
SetCycleNumber(
   [in]    0:UInt64          CycleNumber);
```

Table 7 – SetCycleNumber Method Arguments

Argument	Description
CycleNumber	Number, to which the cycle counter of the LDS shall be set. The next *LDSCycleParametersEvent* will use this value.

Table 8 – SetCycleNumber Method AddressSpace Definition

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

9.8 MaterialBalanceSystemType

Type of the material balance system.

Table 9 – MaterialBalanceSystemTypeEnumeration

Name	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 10.

Table 10 – Values for DeliveryType

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

A server can provide manufacturer specific values with *EnumValues* ≥ 100.

9.11 DeliveryPressure, DeliveryPressureMeasuringPoint

With the objects *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 *ActualValue*, *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/in²)

The variable *DeliveryPressureMeasuringPoint* represents the position of the pressure sensor used for the *DeliveryPressure*. 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 11.

Table 11 – 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

A server can provide manufacturer specific values with *EnumValues* ≥ 100.

NOTE: The actual pressure for each component (for monitoring) is included in the *ComponentType* and the cyclic events.

9.12 DeliveryFlowrate

For system with delivery volumetric flow rate the components *ActualValue*, *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 MixingRatioTarget

Target of the mixing ratio (includes ratio change when MaterialBalanceSystem is active). The share of component A (in percent) defines the value:

Examples: 50 (A 50 : 50 B) → without MaterialBalanceSystem

51,25 (A 51,25 : 48,75 B) → active MaterialBalanceSystem

9.17 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.

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)

The values are given related to the mixing ration of component A. If the maximum allowed mixing ration is 51% component A and 49% component B *MaxDeviationMixingRatio* is 1%.

9.18 RemainingMaterialTime

Remaining time until first material is empty.

9.19 PurgeMode

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

Table 12 – 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

9.20 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 13 – PurgeStatusEnumeration

Name	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.

9.21 ActivateRemoteControl

It is necessary to synchronize the dosing between IMM and LSR dosing systems. This can be done via a separate interfaces e.g. via hardwired signals or also via OPC UA (if the process is robust against small time delays that can be caused by the client/server-connection)

With *ActivateRemoteControl* the client selects the method of remote control. If the server provides only on method for remote control, the other one is not listed in the possible values of the *MultiStateValueDiscreteType*.

Table 14 – Values for ActivateRemoteControl and RemoteControlActivated

EnumValue	ValueAsText	Description
0	OFF	Remote control / automatic mode switched off.
1	SEPARATE_INTERFACE	Activating automatic mode on LDS and using a separate interface from the injection moulding machine for remote control
2	OPC_UA	Activating automatic mode on LDS and using this OPC UA connection with the method StartDosing for remote control

A server can provide manufacturer specific values with *EnumValues* ≥ 100.

9.22 RemoteControlActivated

With this signal, the LDS signalizes, if it is ready to be controlled via this or a separate interface. See Table 14 for possible values.

9.23 StartDosing, StopDosing, DosingActive

Description: If *RemoteControlActivated* = 2, the two *Methods* (without arguments) are used to start and stop the dosing. With the Variable *DosingActive* the LDS can inform the IMM, if dosing is really active.

NOTE: The dosing can also be stopped by the LDS itself (e.g. when SetShotWeight has been reached) to avoid evelasting dosing when the client does not call the method StopDosing.

Signatures:

```
StartDosing();
StopDosing();
```

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

Table 15 – StartDosing Method AddressSpace Definition

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

Table 16 – StopDosing Method AddressSpace Definition

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

9.24 LDSCycleParametersEventType

The *LDSCycleParametersEventType* represents information on a dosing cycle. A complete dosing is defined from the beginning of the dosing signal to the next one, i.e. the event for cycle n is fired, when the dosing signal for cycle n+1 starts.

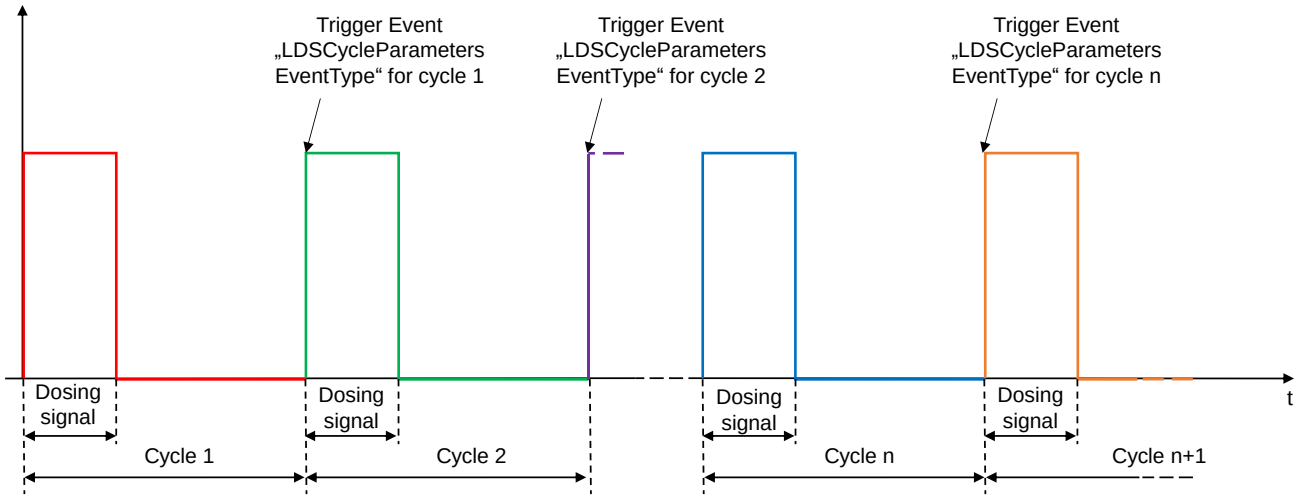

Figure 2 – Timing of LDSCycleParametersEventTyps

The *LDSCycleParametersEventType* is formally defined in Table 17.

Table 17 – LDSCycleParametersEventType Definition

Attribute	**Value**				
BrowseName	LDSCycleParametersEventType				
IsAbstract	True				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:*BaseEventType* defined in OPC UA Part 5					
0:HasProperty	Variable	CycleNumber	0:UInt64	0:PropertyType	M
0:HasProperty	Variable	DosingTime	0:Duration	0:PropertyType	O
0:HasComponent	Variable	MixingRatioTarget	0:Double	0:AnalogItemType	O
0:HasComponent	Variable	MixingRatioActual	0:Double	0:AnalogItemType	O
0:HasComponent	Variable	AdditivesRatioTarget	0:Double[]	0:AnalogItemType	O
0:HasComponent	Variable	AdditivesRatioActual	0:Double[]	0:AnalogItemType	O
0:HasComponent	Variable	VolumeA	0:Double	0:AnalogItemType	O
0:HasComponent	Variable	VolumeB	0:Double	0:AnalogItemType	O
0:HasComponent	Variable	VolumeAB	0:Double	0:AnalogItemType	O
0:HasComponent	Variable	VolumeAdditives	0:Double[]	0:AnalogItemType	O
0:HasComponent	Variable	VolumeTotal	0:Double	0:AnalogItemType	O
0:HasComponent	Variable	ResidualAmountA	0:Double	0:AnalogItemType	O
0:HasComponent	Variable	ResidualAmountB	0:Double	0:AnalogItemType	O
0:HasComponent	Variable	MixingPointPressureA	0:Double	0:AnalogItemType	O
0:HasComponent	Variable	MixingPointPressureB	0:Double	0:AnalogItemType	O
0:HasComponent	Variable	MixingPointPressureBlender	0:Double	0:AnalogItemType	O
0:HasComponent	Variable	AdditivesPressure	0:Double[]	0:AnalogItemType	O
0:HasComponent	Variable	FilterPressurePrimary	0:Double	0:AnalogItemType	O
0:HasComponent	Variable	FilterPressureSecondary	0:Double	0:AnalogItemType	O

9.24.1 Cycle Number

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

Example: 900

9.24.2 DosingTime

Duration of the dosing cycle.

9.24.3 MixingRatioTarget

Target mixing ratio of the last cycle (includes ratio change when MaterialBalanceSystem is active). The share of component A (in percent) defines the value:

Examples: 50 (A 50 : 50 B) → without MaterialBalanceSystem

51,25 (A 51,25 : 48,75 B) → active MaterialBalanceSystem

9.24.4 MixingRatioActual

Actual mixing ratio of the components. The share of component A defines the value:

Example: 50,9 (A 50,9 : 49,1 B)

9.24.5 AdditivesRatioTarget

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

9.24.6 AdditivesRatioActual

Actual ratios of additive in percentage.

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

9.24.7 VolumeA

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

Unit: cm^3 or in^3

9.24.8 VolumeB

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

Unit: cm^3 or in^3

9.24.9 VolumeAB

Volume of components A + B that was added to the process in the last cycle.

Unit: cm^3 or in^3

9.24.10 VolumeAdditives

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

Unit: cm^3 or in^3

9.24.11 VolumeTotal

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

Unit: cm^3 or in^3

9.24.12 ResidualAmountA

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

Unit: kg or lb

9.24.13 ResidualAmountB

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

Unit: kg or lb

9.24.14 MixingPointPressureA

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

Unit: bar or psi

9.24.15 MixingPointPressureB

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

Unit: bar or psi

9.24.16 MixingPointPressureBlender

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

Unit: bar or psi

9.24.17 AdditivesPressure

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

Unit: bar or psi

9.24.18 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

This *ObjectType* contains information about the mixing components A and B. It is formally defined in Table 18.

Table 18 – ComponentType Definition

Attribute	**Value**				
BrowseName	ComponentType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:*BaseObjectType* defined in OPC UA Part 5					
0:HasComponent	Variable	SetValueDensity	0:Double	0:AnalogItemType	O, RO
0:HasComponent	Method	SetSetValueDensity			O
0:HasComponent	Variable	ActualPressure	0:Double	0:AnalogItemType	O, RO
0:HasComponent	Variable	ActualFollowerPlatePressure	0:Double	0:AnalogItemType	O, RO
0:HasComponent	Variable	SetFollowerPlatePressure	0:Double	0:AnalogItemType	O, RW
0:HasComponent	Variable	DrumCapacity	0:Double	0:AnalogItemType	O, RO
0:HasComponent	Variable	ResidualAmount	0:Double	0:AnalogItemType	O, RO
0:HasComponent	Variable	RemainingMaterialTime	0:Duration	0:BaseDataVariableType	O, RO
0:HasProperty	Variable	AllowsCycles	0:Double	0:PropertyType	O, RO
0:HasProperty	Variable	Status	ComponentStatus Enumeration	0:PropertyType	M, RO

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:

```
SetSetValueDensity (
[in]    0:Double         Density);
```

Table 19 – SetSetValueDensity 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 20 – SetSetValueDensity Method AddressSpace Definition

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

10.3 ActualPressure

Actual pressure of the component (between drum and blender measured after the pump).

Unit: bar or psi

10.4 ActualFollowerPlatePressure

Actual material pressure under the follower plate (e.g. calculated by the pressure transmission ratio of pneumatics cylinder to follower plate).

Unit: bar or psi

10.5 SetFollowerPlatePressure

Set value for material pressure under the follower plate (e.g. calculated by the pressure transmission ratio of pneumatics cylinder to follower plate).

Unit: bar or psi

10.6 DrumCapacity

Maximum capacity of the drum (typically 20 or 200 kg).

Unit: kg or lb

10.7 ResidualAmount

Residual amount of the material.

Unit: kg or lb

10.8 RemainingMaterialTime

Time until the material of the component is empty.

10.9 AllowsCycles

Expected number of remaining cycles with the current drum.

10.10 Status

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

Detailed information may be published via *ComponentAlarmType*.

Table 21 – ComponentStatusEnumeration

Name	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

This *ObjectType* contains information about used additives. It is formally defined in Table 22.

Table 22 – AdditiveType Definition

Attribute	Value				
BrowseName	AdditiveType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:*BaseObjectType* defined in OPC UA Part 5					
0:HasProperty	Variable	IsPresent	0:Boolean	0:PropertyType	M, RO
0:HasProperty	Variable	ActivateAdditive	0:Boolean	0:PropertyType	M, RW
0:HasProperty	Variable	AdditiveActivated	0:Boolean	0:PropertyType	M, RO
0:HasProperty	Variable	Status	AdditiveStatus Enumeration	0:PropertyType	M, RO
0:HasComponent	Object	AdditiveFraction		3:ControlledParameterType	O
0:HasComponent	Object	AdditiveStrokeVolume		3:ControlledParameterType	O
0:HasProperty	Variable	ActivateClosedLoopControl	0:Boolean	0:PropertyType	O, RW
0:HasProperty	Variable	ClosedLoopControlActivated	0:Boolean	0:PropertyType	O, RO

11.1 IsPresent

This *Property* informs the client if the additive is physically present. May be FALSE e.g. if the server does not create dynamically instances and has a fixed number of instances for the additives out of which some are currently not used.

11.2 ActivateAdditive

Set value to activate the additive.

11.3 AdditiveActivated

Is *true* if the additive is activated.

11.4 Status

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

Detailed information may be published via *AdditiveAlarmType*.

Table 23 – AdditiveStatusEnumeration

Name	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*).

11.5 AdditiveFraction

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

11.6 AdditiveStrokeVolume

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: mm^3 or in^3

11.7 ActivateClosedLoopControl

Activate the closed loop control of the additive.

11.8 ClosedLoopControlActivated

Is *false* if the closed loop control of the additive is not activated. In this case the ActualValues of AdditiveFraction and AdditiveVolume are equal to the SetValues.

Is *true* if the closed loop control of the additive is activated. In this case, the ActualValues of AdditiveFraction and AdditiveVolume are the measured values.

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 *ComponentAlarmType*, additive-related messages should be represented by instances of *AdditiveAlarmType*, 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 ComponentAlarmType

The *ComponentAlarmType* 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 24 – ComponentAlarmType Definition

Attribute	Value				
BrowseName	ComponentAlarmType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Modelling Rule**
Subtype of 3:*HelpOffNormalAlarmType* defined in OPC 40083					

The *SourceNode* (included in *BaseEventType*) shall contain 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 AdditiveAlarmType

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

Table 25 – AdditiveAlarmType Definition

Attribute	Value				
BrowseName	AdditiveAlarmType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 3:*HelpOffNormalAlarmType* defined in OPC 40083					

The *SourceNode* (included in *BaseEventType*) shall contain 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 Conformance Units

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 26 lists all Profiles defined in this document and defines their URIs.

Table 26 – Profile URIs for OPC 40082-3

Profile	URI
OPC 40082-3 Basic Server Profile	http://opcfoundation.org/UA-Profile/PlasticsRubber/HotRunner/Server/Basic
OPC 40082-3 Alarms Server Facet	http://opcfoundation.org/UA-Profile/PlasticsRubber/HotRunner/Server/Alarms

Table 27 – OPC 40082-3 Basic Server Profile Definition

Conformance Unit	Description	Optional/ Mandatory
OPC 40082-3 Basic Server Profile	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 28 – OPC 40082-3 Alarms Server Facet Definition

Conformance Unit	Description	Optional/ Mandatory
OPC 40082-3 Alarms Server Facet	Support of *HelpOffNormalAlarmType*, *ComponentAlarmType* and *AdditiveAlarmType* 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

14 Namespaces

14.1 Namespace Metadata

Table 29 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 29 – NamespaceMetadata Object for this Specification

Attribute	Value	
BrowseName	http://opcfoundation.org/UA/PlasticsRubber/LDS/	
Property	**DataType**	**Value**
NamespaceUri	String	http://opcfoundation.org/UA/PlasticsRubber/LDS/
NamespaceVersion	String	RC 1.01.0
NamespacePublicationDate	DateTime	2022-07-08
IsNamespaceSubset	Boolean	False
StaticNodeIdTypes	IdType []	0
StaticNumericNodeIdRange	NumericRange []	
StaticStringNodeIdPattern	String	

14.2 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 document shall not use the standard namespaces.

Table 30 provides a list of mandatory and optional namespaces used in an OPC 40082-3 OPC UA *Server*.

Table 30 – Namespaces used in an OPC 40082-3 Server

NamespaceURI	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 31 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 31 – Namespaces used in this specification

NamespaceURI	Namespace Index	Example
http://opcfoundation.org/UA/	0	0:NodeVersion
http://opcfoundation.org/UA/DI/	2	2:DeviceClass
http://opcfoundation.org/UA/PlasticsRubber/GeneralTypes/	3	3:MachineInformationType

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/1.01.0/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/1.01.0/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

_____