VDMA 40001-2

ICS 25.020; 35.240.50

OPC UA for Machinery –
Part 2: Process Values

OPC UA für Machinery –
Teil 2: Prozesswerte

VDMA 40001-2:2023-05 is identical with OPC 40001-2 (Release 1.00)

Document comprises 31 pages

VDMA

Contents

Figures

Tables

OPC Foundation / VDMA

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.

Forewords

Mechanical engineering is a broad-based industry, which is mainly associated with machines such as machine tools, woodworking machines or robots. Many other products such as measuring and testing equipment are also relevant to this field.

Since the information models in this document are intended to apply not only to machines (see in ISO 12100:2010, [1]), but to all other applications and products in the entire machinery industry. Each case cannot be represented individually, therefore the term "machine" is used uniformly for all in this document.

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

Version	**Changes**
OPC 40001 – 2 1.00 (identical with VDMA 40001-2:2023-05)	Initial release

This specification was created by a joint working group of the OPC Foundation and VDMA.

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

VDMA

The VDMA is Europe's largest industry association with over 3300 member companies of the mechanical engineering industry. These companies integrate the latest technologies in products and processes. VDMA was founded in November 1892 and is the most important voice for the mechanical engineering industry today. With the headquarters located in Frankfurt, it represents the issues of the mechanical and plant engineering sector in Germany and Europe. The standard OPC UA has established itself in this industry sector. The VDMA defines OPC UA Companion Specifications for various sectors of the mechanical engineering industry, with more than 450 companies involved. Consequently, one of the main tasks is to harmonise and create consistency.

1 Scope

The OPC UA for Machinery specification contains various building blocks for Machinery that allow to address use cases across different types of machines and components of machines defined in various companion specifications.

For the general scope of the OPC UA for Machinery specification see OPC 40001-1.

This part contains a building block for

- Process Values

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 and errata) applies.

OPC 10000-1, *OPC Unified Architecture - Part 1: Overview and Concepts*

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

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

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

OPC 10000-4, *OPC Unified Architecture - Part 4: Services*

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

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 40001-1, *OPC UA for Machinery - Part 1: Basic Building Blocks*

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

OPC 30081, *OPC UA for Process Automation Devices – PA-DIM™*

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

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 Machinery – Process Values Information Model. For the purposes of this document, the terms and definitions given in OPC 10000-1, OPC 10000-3, OPC 10000-4, OPC 10000-5, OPC 10000-7, OPC 10000-9, OPC 40001-1, as well as the following apply.

Note that OPC UA terms and terms defined in this specification are *italicized* in the specification.

3.2 OPC UA for Process Values terms

No additional terms defined in this specification.

3.3 Abbreviated terms

MES Manufacturing Execution System

PLC Programmable Logic Controller

3.4 Conventions used in this document

For conventions used in this document see OPC 40001-1.

4 General information to Machinery and OPC UA

For general information to Machinery and OPC UA see OPC 40001-1.

5 Use cases

The purpose of this specification is to provide the mechanisms related to the representation of process values for domain-specific Companion Specifications. Accordingly, a number of use cases must be satisfied in this model:

1. The user would like to access the process values of a machine and its various meta data like ranges, precision and unit.
2. The user would like to access and set the setpoints of the process values of a machine.
3. The user would like to access and set deviation limits of the process values, relative to the setpoints.
4. The user would like to get informed when a process value is passing a deviation limit or range.
5. The user would like to get the percentage value of a process variable, also when there are dynamic ranges.
6. The user would like to zero-point adjust the current value of a process value.
7. The user would like to get vendor-specific error codes on devices providing process values.
8. The user would like to access and set a substitution value in case of connections lost.
9. The user would like to get identification information of devices providing process values.
10. The user would like to get information about the health status of devices providing process values.

6 Process Values Information Model overview

6.1 General

This information model provides information about process values, for example provided by actuators or sensors. In 6.2, the integration of such devices as components of a machine is described. This includes information about the identification of a device as well as health information and specific errors. In 9.1, the *VariableType* representing process value setpoints is introduced. In 6.4, an *ObjectType* using the *VariableType* and providing the process value as well as adding additional mechanisms for alarming and zero-point adjustment is described. Examples of the overall usage of process values are described in 6.5.

6.2 Integration as Device in Machinery Specification

In order to integrate process values in the Machinery Information Model including device information, the device should be represented as component of a machine. An example *ObjectType* X:MySensorType is shown in Figure 1. It provides identification information using the 4:*MachineryComponentIdentificationType*. The process values – a device can potentially provide more than one process value – are grouped by the 3:*SignalSet* defined in the 3:*ISignalSetType* of OPC 30081. To provide the overall health status of the device as well as specific errors, the 2:*IDeviceHealthType* is implemented, defined in OPC 10000-100.

Note that it is not required to provide any device information to use the process values defined in this specification. However, this mechanism provides information about the identification and health status of the device.

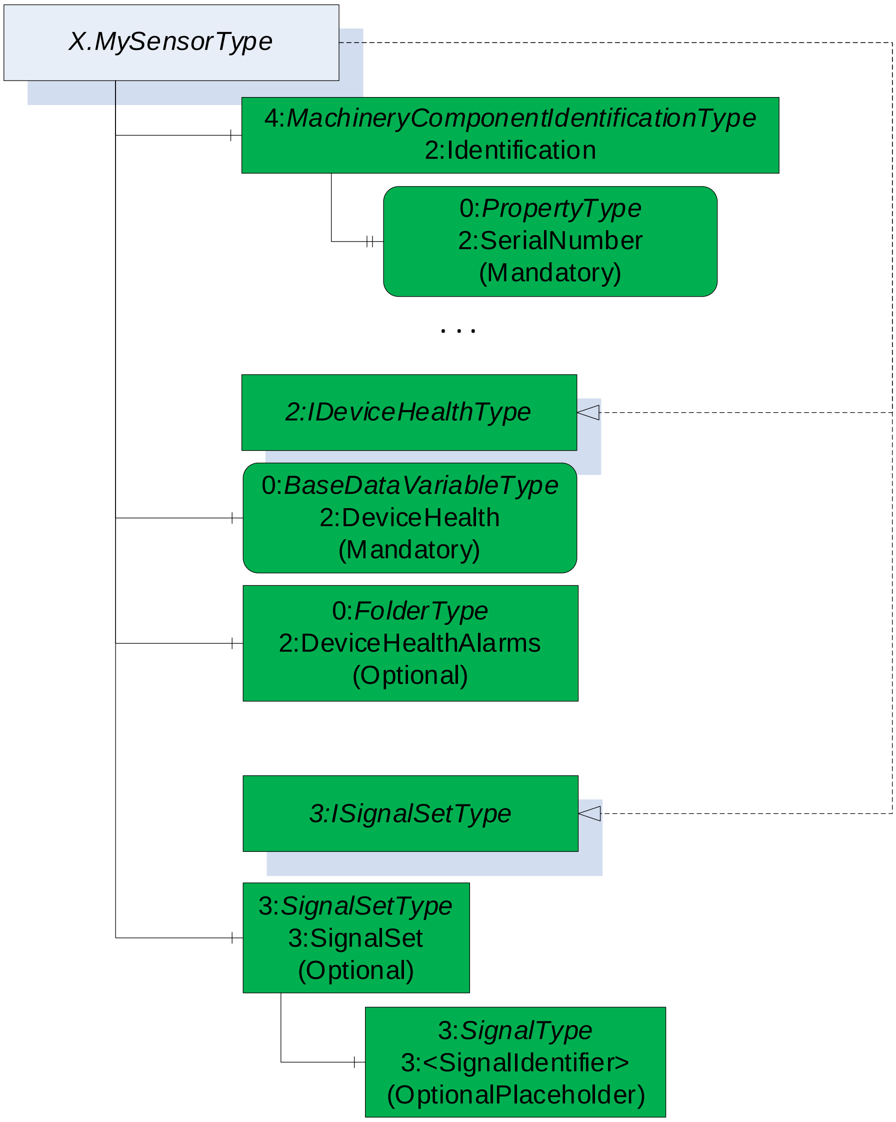

Figure 1 – Integration as Device in Machinery Specification

6.3 ProcessValueSetpointVariableType

In Figure 2, illustrates the 3:*AnalogSignalVariableType* and the *ProcessValueSetpointVariableType* and their supertypes with key *InstanceDeclarations*. The 3:*AnalogSignalVariableType* provides the optional *Properties* 0:*ValuePrecision* and 0:*InstrumentRange* and the mandatory *Properties* 0:*EURange* and 0:EngineeringUnits and are used as defined in OPC 10000-8. They provide information about the precision, ranges and unit of the process value. The optional *Variables* 3:*ActualValue*, 3:*SimulationValue*, and 3:*SimulationState* are used as defined in OPC 30081 and provide a simulation value and a flag to define, which value is used.

The *ProcessValueSetpointVariableType* defined in 9.1 adds additional, optional meta data.

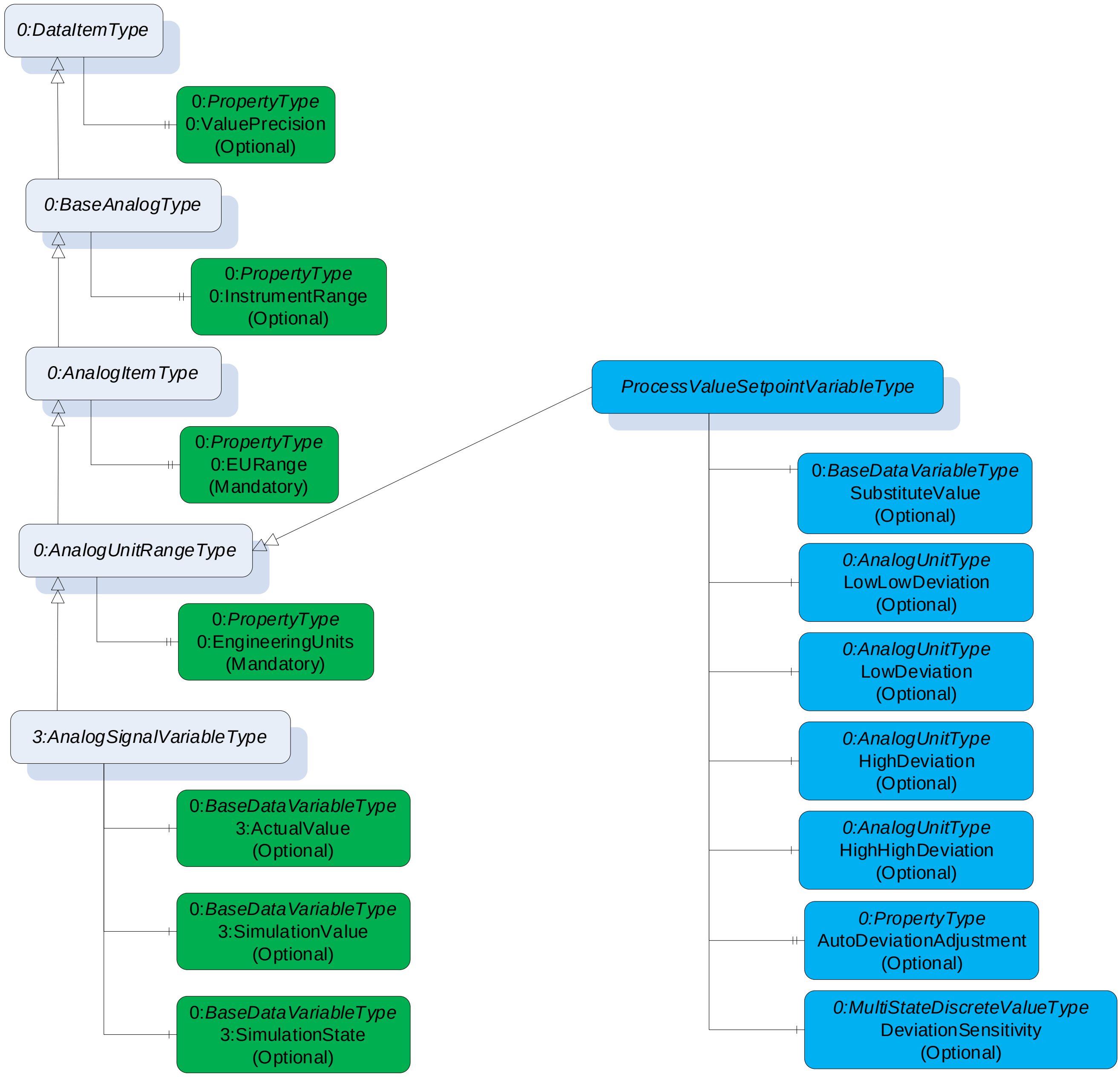

Figure 2 – ProcessValueSetpointVariableType

6.4 ProcessValueType

In Figure 3, the *ProcessValueType* and its supertypes with key *InstanceDeclarations* is shown. The mandatory 3:*SignalTag* contains a unique name as defined by OPC 30081. The optional 3:*ZeroPointAdjustment Method* allows to set a zero point as defined in OPC 30081. The *ProcessValueType* adds a reference to the *ZeroPointAdjustmentEventType*. *Events* of this *EventType* are generated, when the zero-point adjustment is executed (see 8.1). The 3:*AnalogSignal* defined in OPC 30081 is extended with additional meta data. The *ProcessValueType* is defined in 7.1, having additional optional components like process value setpoint and alarm information.

This *ObjectType* was introduced to bind functionality to the process value that cannot be added to the *3:AnalogSignal* directly, like *Objects* and *Methods*.

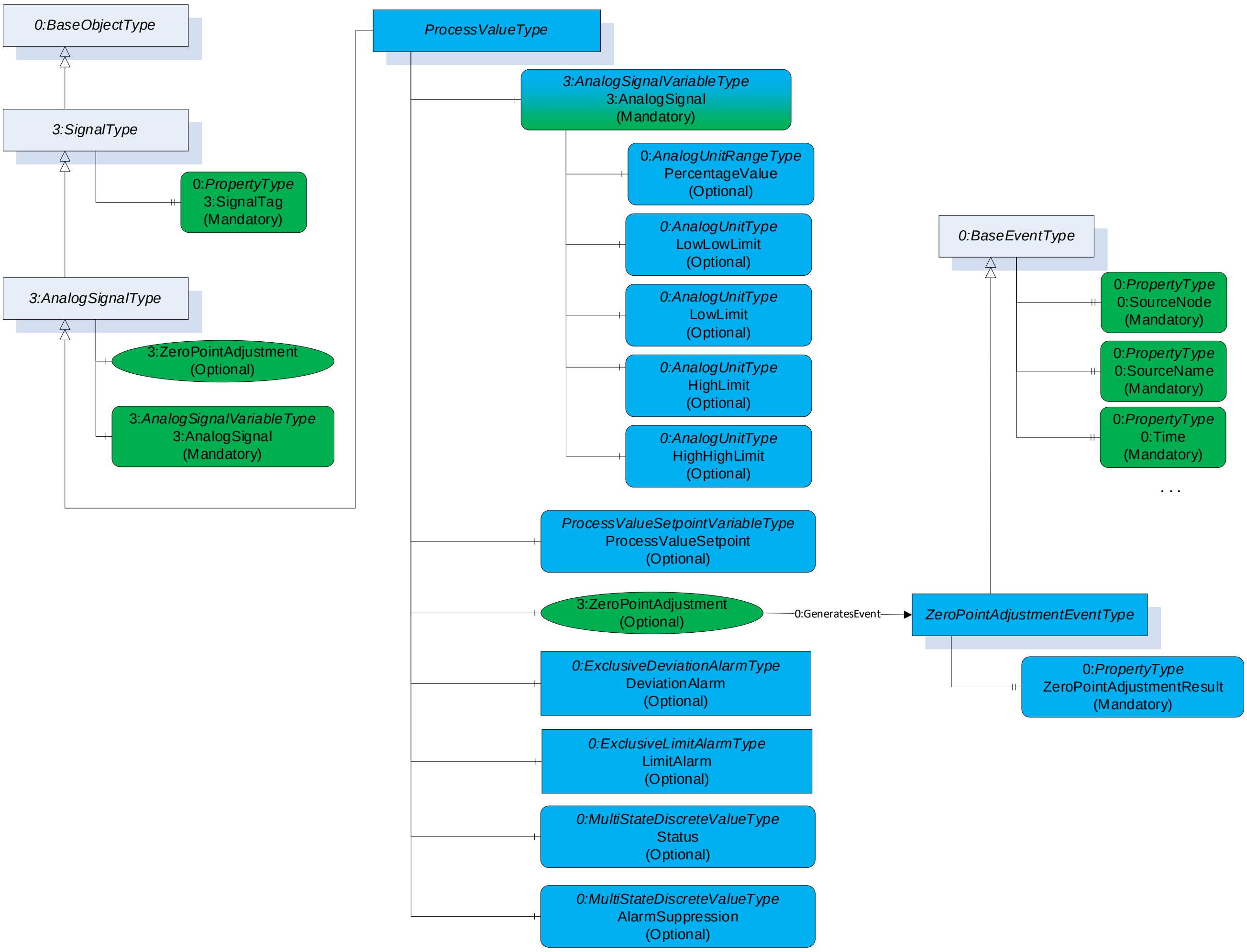

Figure 3 – ProcessValueType

6.5 Examples applying Process Values

In Figure 4, an example of applying the device values in a machine is given, providing all information defined in this specification. The X:*MyMachine Object* contains a X:*Sensor1* in its 4:*Components* folder. The X:*Sensor1* provides identification information and health status. In addition, it contains two process values, X:*Temperature* and X:*Pressure*. Those *Objects* can be referenced from other paths as well, like the X:*Monitoring Object* of X:*MyMachine*. The X:*Pressure Object* contains various capabilities like the 3:*ZeroPointAdjustment Method*, *DeviationAlarm*, and the 3:*AnalogSignal* containing the process value. This *Variable* contains various sub-variables with meta data for ranges and unit, deviation etc.

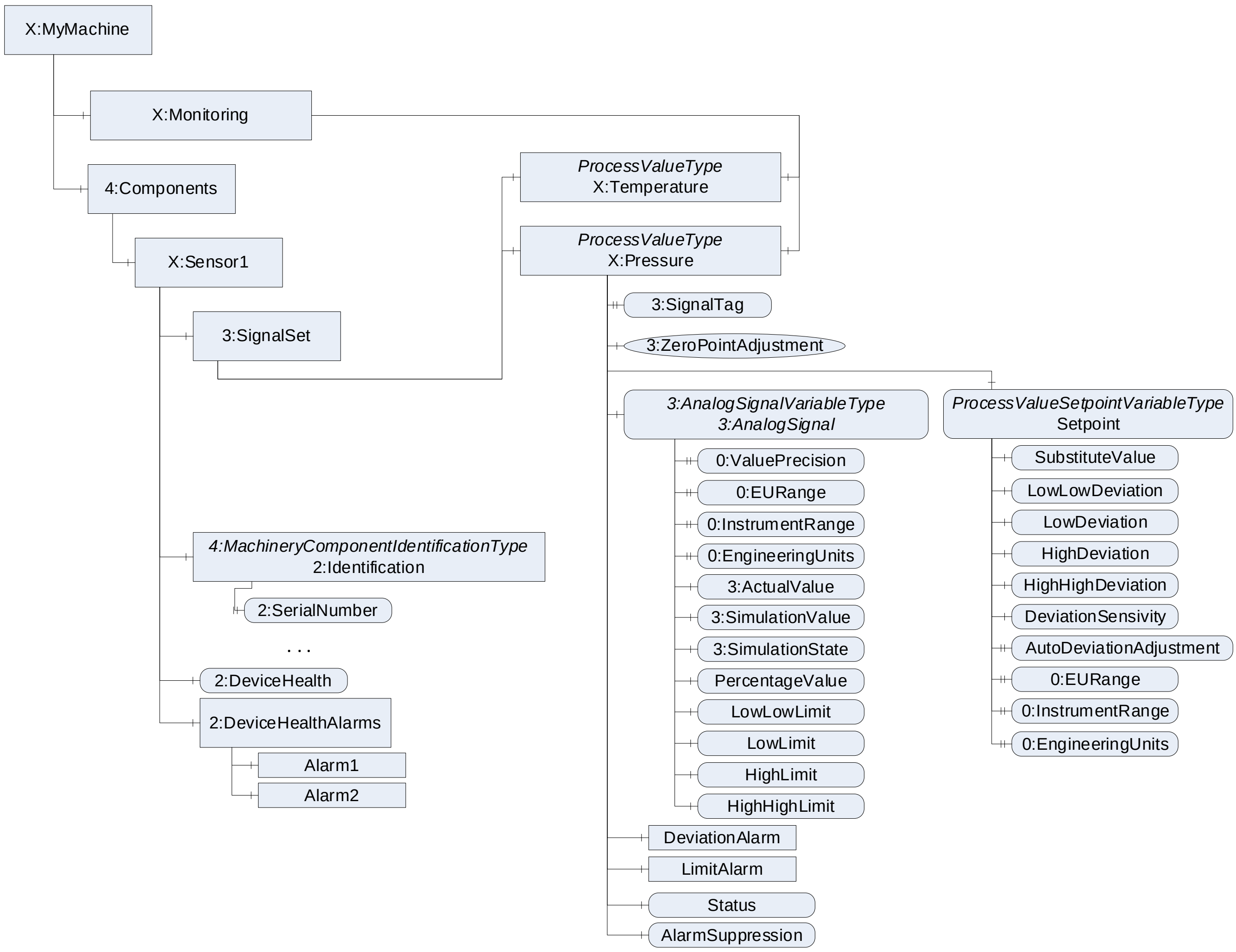

Figure 4 – Example applying Device Values with all information

In Figure 5, an example is given using the base infrastructure defined in this specification to provide process values. In this case, no device information or the optional information is given, the X:MyMachine just provided some process values under the X:Monitoring *Object*.

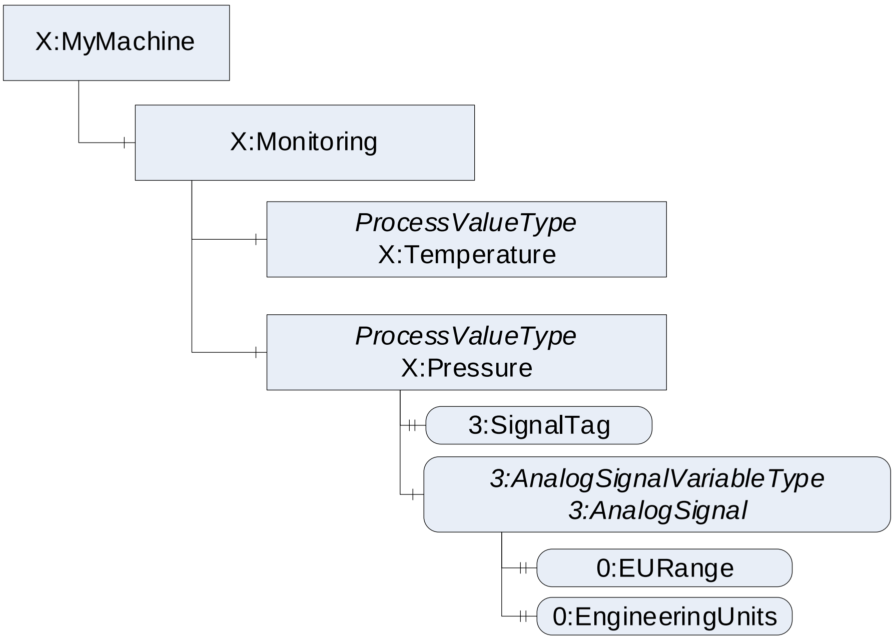

Figure 5 – Example applying Device Values with base information

6.6 Defining Setpoints without an associated Process Value

The *ProcessValueType* is designed to represent a process value having optionally a process value setpoint. It is not designed to represent a setpoint without a process value. If a setpoint without an associated process values should be represented, using the *ProcessValueType* is inappropriate. Information models may use the *ProcessValueSetpointVariableType* as representation of a setpoint. But also this *VariableType* has optional subvariables that imply a process value (everything associated to deviation) that should not be used without an associated process value.

7 OPC UA ObjectTypes

7.1 ProcessValueType ObjectType Definition

The *ProcessValueType* represents a process value. It is formally defined in Table 1.

Table 1 – ProcessValueType Definition

Attribute	**Value**				
BrowseName	ProcessValueType				
IsAbstract	False				
Description	Represents a process value				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 3:AnalogSignalType defined in OPC 30081					
0:HasComponent	Variable	3:AnalogSignal	0:Number{Any}	3:AnalogSignalVariableType	M
0:HasComponent	Variable	ProcessValueSetpoint	0:Number{Any}	ProcessValueSetpointVariableType	O
0:HasComponent	Method	3:ZeroPointAdjustment			O
0:HasComponent	Object	DeviationAlarm		0:ExclusiveDeviationAlarmType	O
0:HasComponent	Object	LimitAlarm		0:ExclusiveLimitAlarmType	O
0:HasComponent	Variable	Status	0:UInt16	0:MultiStateValueDiscreteType	O
0:HasComponent	Variable	AlarmSuppression	0:UInt16	0:MultiStateValueDiscreteType	O
Conformance Units					
Machinery Process Values Base Types					

This *ObjectType* inherits the *InstanceDeclarations* defined on its supertypes, including the mandatory 3:*SignalTag* as defined in OPC 30081.

The mandatory 3:*AnalogSignal Variable* is overridden and additional *InstanceDeclarations* are added (see Table 2). It contains the process value and meta data about the process value.

The 3:*AnalogSignalVariableType* already provides the mandatory 0:*EURange*, 0:*EngineeringUnits* and the optional 0:*ValuePrecision*, 0:*InstrumentRange*, 3:*ActualValue*, 3:*SimulationValue* and 3:*SimulationState* as defined in OPC 10000-8 and OPC 30081.

The *PercentageValue Variable* on the 3:*AnalogSignal* provides the process value in percentage. The 0:*EngineeringUnits* of the *Variable* shall always be percentage (UnitId: 20529 with NamespaceUri http://www.opcfoundation.org/UA/units/un/cefact). This *Variable* is especially useful when the ranges for calculating the percentage values are dynamic, so that the client cannot calculate the percentage based on the 0:*EURange* of the *3:AnalogSignal* Variable. As an example, the wear of a filter should be exposed, by providing the differential pressure. The pressure is presented as the *3:AnalogSignal* and the wear is presented as the *PercentageValue.* Suction output is an internal value. However, the differential pressure depends on the suction output. With 50% suction output the differential pressure of 250 Pa is 0% wear (meaning new filter), for 100% suction output it is 500 Pa and 0% wear. For a used filter, the values are 1600 Pa and 60% wear for 50% suction output and 1700 Pa and 60% wear for 100% suction output. A filter to be replaced would have 2500 Pa and 100% wear for 50% suction output and 2500 Pa and 100% wear for 100% suction output.

There are four optional limit *Variables* on 3:*AnalogSignal*: *LowLowLimit*, *LowLimit*, *HighLimit* and *HighHighLimit*. They define different limits for the *Value* of the *Variable,* either absolute or in percentage. It is not required to provide all limit *Variables*. For example, a process value may only have a *LowLimit*.

- If the limit is defined in percentage, the 0:*EngineeringUnits* shall be percentage (UnitId: 20529 with NamespaceUri http://www.opcfoundation.org/UA/units/un/cefact). The percentage value is calculated by the range of values between the high and the low limit. 100% therefore corresponds to the equation "EURange.High minus EURange.Low".
- If the limit is defined absolute, the 0:*EngineeringUnits* shall be the same as for the *Variable.*

If one limit *Variable* is defined in percentage, all limit *Variables* shall be defined in percentage.

For all instances of *ProcessValueType* having an 3:*AnalogSignal* providing any of the limit Variables, the 3:*AnalogSignal* shall have scalar *Values*. Subtypes may be created using arrays and defining the expected behaviour with respect to the limit *Variables*.

The limit *Variables* are defined in an order. The *LowLowLimit* shall be smaller or equal *LowLimit*, which shall be smaller or equal *HighLimit*, which shall be smaller or equal *HighHighLimit*.

In Figure 6, the relation of the limit *Variables*, and the ranges is shown. The limit *Variables* are absolute, as well as the ranges.

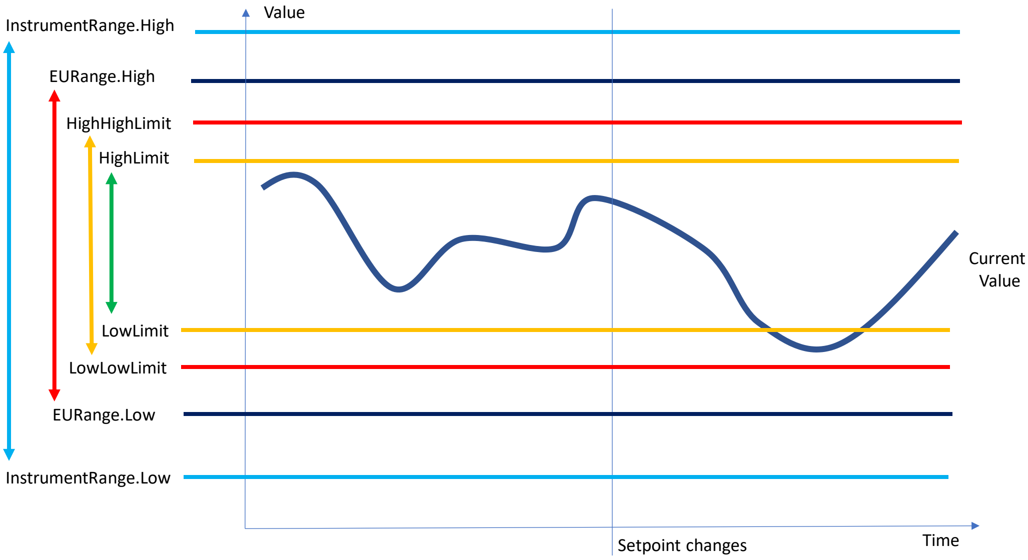

Figure 6 – Relation between limit Variables, ranges, and current Value

Note that *Servers* may generate 0:*LimitAlarms* when the deviation limits are reached. The limit *Variables* are used to configure this alarm.

The optional *ProcessValueSetpoint Variable* provides the desired value of the 3:*AnalogSignal*. The *Server* may, or may not control the *Value* to reach the process value setpoint. The *DataType*, *ValueRank* and *ArrayDimensions* shall be the same as for the 3:*AnalogSignal*. The 0:*EngineeringUnits Property* shall have the same *Value* as the 0:*EngineeringUnits Property* of 3:*AnalogSignal.* The 0:*InstrumentRange Property*, if provided, shall have the same *Value* as the 0:*InstrumentRange Property* of 3:*AnalogSignal.* The 0:*EURange Property* shall have the same *Value* as the 0:*InstrumentRange Property* of 3:*AnalogSignal,* or further restrict the range, i.e. it may have a larger low and a smaller high value.

The optional 3:*ZeroPointAdjustment Method* is overridden and works as defined in OPC 30081. The *ProcessValueType* adds a 0:*GeneratesEvent Reference* to the *ZeroPointAdjustmentEventType*. When the *Method* is called and the execution of the *Method* starts, a corresponding *Event* is generated. Such an Event shall not be generated, when the Method was called but execution was not started, for example due to access restrictions or invalid input arguments.

The optional *DeviationAlarm Object* becomes active, when the process value deviates from the *ProcessValueSetpoint*. The limits of the *DeviationAlarm* are bound to the deviation *Variables* of the *ProcessValueSetpoint*. As the deviation *Variables* of the *ProcessValueSetpoint* may be defined in percentage the limits of the DeviationAlarm may need to be calculated accordingly. The 0:*LowLowLimit* shall be mapped to *LowLowDeviation*, 0:*LowLimit* to *LowDeviation*, 0:*HighLimit* to *HighDeviation* and 0:*HighHighLimit* to *HighHighDeviation*. If a deviation *Variable* is not present, the corresponding limit shall not be provided as well. If no deviation *Variable* is present, the *DeviationAlarm* shall not be provided.

The optional *LimitAlarm Object* becomes active, when the process value reaches a limit. The limits of the *LimitAlarm* are bound to the limit *Variables* of the 3:*AnalogSignal*. As the limit *Variables* of the *ProcessValueSetpoint* may be defined in percentage the limits of the DeviationAlarm may need to be calculated accordingly. The 0:*LowLowLimit* shall be mapped to *LowLowLimit*, 0:*LowLowLimit* to *LowLimit,* 0:*HighLimit* to *HighLimit* and 0:*HighHighLimit* to *HighHighLimit*. If a limit *Variable* is not present, the corresponding limit shall not be provided as well. If no limit *Variable* is present, the *LimitAlarm* shall not be provided.

The optional *Status Variable* indicates if a deviation limit or an absolute limit has been reached. This specification defines standardized 0:*EnumValues* (see Table 3) that shall be used as defined in this specification or omitted.

Servers may define additional 0:*EnumValues* starting with 256. The deviation *Variables* uses the deviation *Variables* of the *ProcessValueSetpoint*, and the limit *Variables* of the 3:*AnalogSignal*.

For the standardized 0:*EnumValues* there is a priority, which value shall be reported, when several limits are reached. If the HighHighLimit or the LowLowLimit is reached, it shall be reported. If those are not reached, but the HighLimit or LowLimit is reached, it shall be reported. If those are not reached, but the HighHighDeviation or LowLowDeviation is reached, it shall be reported. If those are not reached, but the HighDeviation or LowDeviation is reached, it shall be reported. If there are additional 0:*EnumValues* defined, the priority of those is not defined, and they could be reported instead of the standardized ones.

The optional *AlarmSuppression Variable* indicates if alarms based on the *Status* shall be suppressed. This might be useful for example when starting a machine Alarms include but are not limited to OPC UA Alarms, they can for example also be acoustic or visual indications on the machine. This specification defines standardized 0:*EnumValues* (see Table 3) that shall be used as defined in this specification or omitted. *Servers* may define additional 0:*EnumValues* starting with 256.

The components of the *ProcessValueType* have additional subcomponents which are defined in Table 2.

Table 2 – ProcessValueType Additional Subcomponents

BrowsePath	References	NodeClass	BrowseName	DataType	TypeDefinition	Others
3:AnalogSignal	0:HasComponent	Variable	PercentageValue	0:Double	0:AnalogUnitRangeType	O
3:AnalogSignal	0:HasComponent	Variable	LowLowLimit	0:Number	0:AnalogUnitType	O
3:AnalogSignal	0:HasComponent	Variable	LowLimit	0:Number	0:AnalogUnitType	O
3:AnalogSignal	0:HasComponent	Variable	HighLimit	0:Number	0:AnalogUnitType	O
3:AnalogSignal	0:HasComponent	Variable	HighHighLimit	0:Number	0:AnalogUnitType	O

The child *Nodes* of the *ProcessValueType* have additional *Attribute* values defined in Table 3.

Table 3 – ProcessValueType Attribute values for child Nodes

BrowsePath	Value Attribute	Description Attribute
3:AnalogSignal	-	The process value.
3:AnalogSignal PercentageValue	-	Provides the process value in percentage.
3:AnalogSignal PercentageValue 0:EngineeringUnits	NamespaceUri: http://www.opcfoundation.org/UA/units/un/cefact UnitId: 20529 DisplayName: % Description: percent	-
3:AnalogSignal PercentageValue 0:EURange	Low: 0 High: 100	-
3:AnalogSignal LowLowLimit	-	Defines the absolute low low limit
3:AnalogSignal LowLimit	-	Defines the absolute low limit
3:AnalogSignal HighLimit	-	Defines the absolute high limit
3:AnalogSignal HighHighLimit	-	Defines the absolute high high limit
ProcessValueSetpoint	-	The desired value, may or may not be controlled by the server.
DeviationAlarm	-	Becomes active, when the process values derivates from the ProcessValueSetpoint.
LimitAlarm	-	Becomes active, when absolute limits are reached.
Status	-	Indicates if a limit has been reached.

Status 0:EnumValues	{{0, NONE, Not monitoring}, {1, UNKNOWN, Status not known}, {2, BELOW_LOWLOW_LIMIT, Value is below low LowLowLimit}, {3, BELOW_LOW_LIMIT, Value is below LowLimit}, {4, BELOW_LOWLOW_DEVIATION, Value is below LowLowDeviation}, {5, BELOW_LOW_DEVIATION, Value is below LowDeviation}, {6, WITHIN_TOLERANCE, Value is in tolerance}, {7, ABOVE_HIGH_DEVIATION, Value is above HighDeviation}, {8, ABOVE_HIGHHIGH_DEVIATION, Value is above HighHighDeviation}, {9, ABOVE_HIGH_LIMIT, Value is above HighLimit}, {10, ABOVE_HIGHHIGH_LIMIT, Value is above HighHighLimit}}	-
AlarmSuppression	-	Indicates if alarms based on the Status shall be suppressed.
AlarmSuppression 0:EnumValues	{{0, OFF, no alarm suppression}, {1, HORN, suppresses only horn}, {2, COMPLETE, all alarms are suppressed}}	-

The components of the *ProcessValueType* have additional references which are defined in Table 4.

Table 4 – ProcessValueType Additional References

SourceBrowsePath	Reference Type	Is Forward	TargetBrowsePath
3:ZeroPointAdjustment	0:GeneratesEvent	True	ZeroPointAdjustmentEventType

8 OPC UA EventTypes

8.1 ZeroPointAdjustmentEventType ObjectType Definition

The *ZeroPointAdjustmentEventType* provides information, that a zero-point adjustment took place. It is formally defined in Table 5.

Table 5 – ZeroPointAdjustmentEventType Definition

Attribute	Value				
BrowseName	ZeroPointAdjustmentEventType				
IsAbstract	True				
Description	Provides information, that a zero-point adjustment took place				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:BaseEventType defined in OPC 10000-5					
0:HasProperty	Variable	ZeroPointAdjustmentResult	0:StatusCode	0:PropertyType	M
Conformance Units					
Machinery Process Values Base EventTypes					

This *EventType* inherits all *Properties* of the *BaseEventType*. The 0:SourceNode and 0:SourceName shall be set to the Object the zero-adjustment is bound to (like instances of the *ProcessValueType*). The 0:*Time* shall be set to when the zero-point-adjustment took place.

The mandatory *ZeroPointAdjustmentResult Variable* reflects if the zero-point adjustment was successful. If it was triggered by a *Method* call, it shall contain the return *StatusCode* of the *Method* call.

9 OPC UA VariableTypes

9.1 ProcessValueSetpointVariableType VariableType Definition

The *ProcessValueSetpointVariableType* is a subtype of 0:*AnalogUnitRangeType*. It is used to define the desired value of the *Variable* it belongs to. It is, for example, used by the *ProcessValueType*, defining the desired value for the 3:*AnalogSignal*. The *DataType*, *ValueRank* and *ArrayDimensions* shall be identical to the *Variable* it belongs to. The 0:*EngineeringUnits*, of that *Variable* shall have the same *Values* as defined on the *Variable* it belongs to. The 0:*EURange* shall be the same, or further restricted by providing a smaller high or a higher low value. Instances of that VariableType shall not provide the 0:*InstrumentRange*. A process value setpoint *Variable* does not require the server to execute any internal logic to bring the process value (*Variable* to which the process value setpoint belongs) to the process value setpoint.

The *VariableType* is formally defined in Table 6.

Table 6 – ProcessValueSetpointVariableType Definition

Attribute	**Value**				
BrowseName	ProcessValueSetpointVariableType				
IsAbstract	False				
ValueRank	-2 (-2 = Any)				
DataType	Number				
Description	Define the desired value of the Variable it belongs to.				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:AnalogUnitRangeType					
0:HasComponent	Variable	SubstituteValue	0:Number{Any}	0:BaseDataVariableType	O
0:HasComponent	Variable	LowLowDeviation	0:Number	0:AnalogUnitType	O
0:HasComponent	Variable	LowDeviation	0:Number	0:AnalogUnitType	O
0:HasComponent	Variable	HighDeviation	0:Number	0:AnalogUnitType	O
0:HasComponent	Variable	HighHighDeviation	0:Number	0:AnalogUnitType	O
0:HasProperty	Variable	AutoDeviationAdjustment	0:Boolean	0:PropertyType	O
0:HasComponent	Variable	DeviationSensitivity	0:UInt16	0:MultiStateValueDiscrete Type	O
Conformance Units					
Machinery Process Values Base Types					
Machinery Process Values Base SetpointType					

The optional *SubstituteValue Variable* provides a value that should be used when the process value setpoint cannot be controlled anymore. This specification does not define, when this is the case. There might be, for example, a connection lost from the controlling device. The *SubstituteValue* shall have the same *DataType*, *ValueRank* and *ArrayDimensions* as the process value setpoint *Variable*. The same meta data as defined for the process value setpoint *Variable* apply.

There are four optional deviation *Variables*: *LowLowDeviation*, *LowDeviation*, *HighDeviation* and *HighHighDeviation*. They define different limits for deviation. The deviation considered is between the value of the process value setpoint and the value of the *Variable* to which the process value setpoint belongs. The deviation limits are defined relative to the process value setpoint. It is not required to provide all deviation *Variables*. For example, a process value may only have a *LowDeviation*. The deviation can either be defined in percentage or absolute.

- If the deviation is defined in percentage, the 0:*EngineeringUnits* shall be percentage (UnitId: 20529 with NamespaceUri http://www.opcfoundation.org/UA/units/un/cefact). The percentage value is calculated by the range of values between the high and the low limit. 100% therefore corresponds to the equation "EURange.High minus EURange.Low".
- If the deviation is defined absolute, the 0:*EngineeringUnits* shall be the same as the process value setpoint.

If one deviation *Variable* is defined in percentage, all deviation *Variables* shall be defined in percentage.

All instances of the *ProcessValueSetpointVariableType* providing any of the deviation *Variables* shall have scalar *Values*. Subtypes may be created using arrays and defining the expected behaviour with respect to the deviation *Variables*.

The deviation *Variables* are defined in an order. The *LowLowDeviation* shall be smaller or equal *LowDeviation*, which shall be smaller or equal zero. *HighHighDeviation* shall be larger or equal *HighDeviation*, which shall be larger or equal to zero.

In Figure 7, the relation of the deviation *Variables*, the *ProcessValueSetpoint*, and the ranges is shown. The deviation *Variables* are relative to the *ProcessValueSetpoint*, whereas the ranges are absolute. In Figure 7, the *ProcessValueSetpoint* changes at some point in time, and therefore the absolute value for the deviation. Note, that a change of the *ProcessValueSetpoint* may lead to the absolute value of a deviation may pass the ranges.

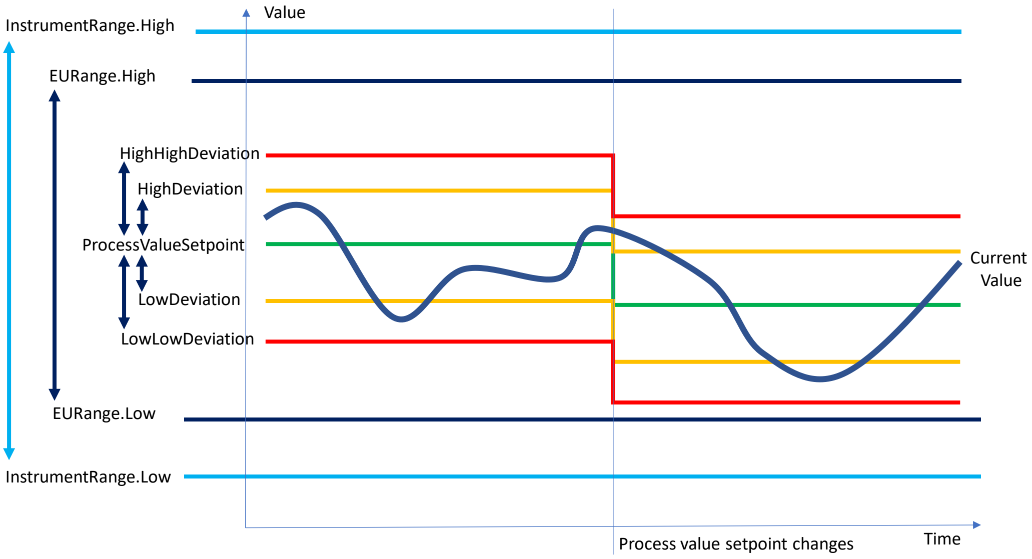

Figure 7 – Relation between deviation Variables, ranges, process value setpoint and current Value

Note that *Servers* may generate 0:*DeviationAlarms* (see 7.1) when the deviation limits are reached. The deviation *Variables* are used to configure this alarm.

The optional *AutoDeviationAdjustment Variable* defines if the deviation *Variables* are automatically adjusted based on the settings of *DeviationSensitivity* (if provided) in case the configuration has changed (0:*EURange* or *DeviationSensitivity*). If set to TRUE, the deviation *Variables* shall be automatically adjusted and writing the deviation *Variables* shall fail.

The optional *DeviationSensitivity Variable* indicates the sensitivity of the deviation *Variables* when automatically set. This specification defines standardized 0:*EnumValues* (see Table 7) that shall be used as defined in this specification or omitted. *Servers* may define additional 0:*EnumValues* starting with 256.

Note that the predefined 0:*EnumValues* (FINE, MIDDLE, and ROUGH) do not define the absolute widths of the set deviation bands, they are device dependent.

The child *Nodes* of the *ProcessValueSetpointVariableType* have additional *Attribute* values defined in Table 7.

Table 7 – ProcessValueSetpointVariableType Attribute values for child Nodes

BrowsePath	Value Attribute	Description Attribute
LowLowDeviation	-	Defines the low low limit for deviation, relative to the process value setpoint.
LowDeviation	-	Defines the low limit for deviation, relative to the process value setpoint.
HighDeviation	-	Defines the high limit for deviation, relative to the process value setpoint.
HighHighDeviation	-	Defines the high high limit for deviation, relative to the process value setpoint.
AutoDeviationAdjustment	-	Defines if the deviation variables are automatically adjusted.
DeviationSensitivity	-	Indicates the sensitivity of the deviation variables when automatically set.

SubstituteValue	-	Value that should be used when the process value setpoint cannot be controlled anymore.
DeviationSensitivity 0:EnumValues	{{0, FINE, tight tolerances}, {1, MIDDLE, mean tolerances}, {2, ROUGH, large tolerances}}	-

10 Profiles and Conformance Units

10.1 Conformance Units

This chapter defines the corresponding *Conformance Units* for the OPC UA Information Model for Machinery – Process Values.

Table 8 – Conformance Units for Machinery – Process Values

Category	Title	Description
Server	Machinery Process Values Base SetpointType	Server exposes the ProcessValueSetpointVariableType and all its supertypes in the AddressSpace.
Server	Machinery Process Values Base Process Value Setpoint	Server is configurable to support at least one instance of ProcessValueSetpointVariableType.
Server	Machinery Process Values Base Types	Server exposes the ProcessValueType and ProcessValueSetpointVariableType and all their supertypes in the AddressSpace.
Server	Machinery Process Values Base EventTypes	Server exposes the ZeroPointAdjustmentEventType and all its supertypes in the AddressSpace.
Server	Machinery Process Values Analog Object Instances	Server is configurable to support at least one instance of ProcessValueType.
Server	Machinery Process Values ZeroPointAdjustment Events	Server is configurable to support at least one instance of ProcessValueType that generates Events of ZeroPointAdjustmentEventType. All instances that support the ZeroPointAdjustment Method generate Events of ZeroPointAdjustmentEventType.
Server	Machinery Process Values Percentage Value	Server is configurable to support at least one instance of ProcessValueType providing the PercentageValue Variable on the AnalogSignal.
Server	Machinery Process Values Deviation Base	Server is configurable to support at least one instance of ProcessValueSetpointVariableType having at least one deviation Variable
Server	Machinery Process Values Monitoring	Server is configurable to support at least one instance of ProcessValueType providing the Status Variable.
Server	Machinery Process Values Deviation AutoAdjustment	Server is configurable to support at least one instance of ProcessValueSetpointVariableType providing the AutoDeviationAdjustment Variable.
Server	Machinery Process Values Deviation Sensitivity	Server is configurable to support at least one instance of ProcessValueSetpointVariableType providing the DeviationSensitivity Variable.
Server	Machinery Process Values Deviation Alarm	Server is configurable to support at least one instance of ProcessValueType providing alarms of the ExclusiveDeviationAlarmType.
Server	Machinery Process Values Deviation Alarm Object	Server is configurable to support at least one instance of ProcessValueType providing the DeviationAlarm Object.
Server	Machinery Process Values AlarmSuppression	Server is configurable to support at least one instance of ProcessValueType providing the AlarmSuppression Variable.
Server	Machinery Process Values Limits Base	Server is configurable to support at least one instance of ProcessValueType having at least one limit Variable on the AnalogSignal.
Server	Machinery Process Values Limits Alarm	Server is configurable to support at least one instance of ProcessValueType providing alarms of the ExclusiveLimitAlarmType.
Server	Machinery Process Values Limits Alarm Object	Server is configurable to support at least one instance of ProcessValueType providing the LimitAlarm Object.
Server	Machinery Process Values Device Object	Server is configurable to support at least one Object providing the MachineryComponentIdentificationType AddIn and implementing the ISignalSet referencing an Object of Type ProcessValueType.

10.2 Profiles

10.2.1 Profile list

Table 9 lists all Profiles defined in this document and defines their URIs.

Table 9 – Profile URIs for Machinery – Process Values

Profile	URI
Machinery-Process Values Base Server Facet	http://opcfoundation.org/UA/Machinery/ProcessValues/Server/Base/
Machinery-Process Values Simple Device Info Server Facet	http://opcfoundation.org/UA/Machinery/ProcessValues/Server/SimpleDeviceInfo/
Machinery-Process Values Zero Point Adjustment Base Server Facet	http://opcfoundation.org/UA/Machinery/ProcessValues/Server/ZeroPointAdjustmentBase/
Machinery-Process Values Zero Point Adjustment Events Server Facet	http://opcfoundation.org/UA/Machinery/ProcessValues/Server/ZeroPointAdjustmentEvents/
Machinery-Process Values Simulation Server Facet	http://opcfoundation.org/UA/Machinery/ProcessValues/Server/Simulation/
Machinery-Process Values Base Process Value Setpoint Server Facet	http://opcfoundation.org/UA/Machinery/ProcessValues/Server/Setpoint/
Machinery-Process Values Percentage Value Server Facet	http://opcfoundation.org/UA/Machinery/ProcessValues/Server/PercentageValue/
Machinery-Process Values Deviation Base Server Facet	http://opcfoundation.org/UA/Machinery/ProcessValues/Server/DeviationBase/
Machinery-Process Values Deviation AutoAdjustment Server Facet	http://opcfoundation.org/UA/Machinery/ProcessValues/Server/DeviationAutoAdjustment/
Machinery-Process Values Deviation Monitoring Server Facet	http://opcfoundation.org/UA/Machinery/ProcessValues/Server/DeviationMonitoring/
Machinery-Process Values Deviation Alarm Server Facet	http://opcfoundation.org/UA/Machinery/ProcessValues/Server/DeviationAlarm/
Machinery-Process Values Deviation Alarm Suppression Server Facet	http://opcfoundation.org/UA/Machinery/ProcessValues/Server/DeviationAlarmSuppression/
Machinery-Process Values Limits Base Server Facet	http://opcfoundation.org/UA/Machinery/ProcessValues/Server/LimitsBase/
Machinery-Process Values Limits Monitoring Server Facet	http://opcfoundation.org/UA/Machinery/ProcessValues/Server/LimitsMonitoring/
Machinery-Process Values Limits Alarm Server Facet	http://opcfoundation.org/UA/Machinery/ProcessValues/Server/LimitsAlarm/
Machinery-Process Values Limits Alarm Suppression Server Facet	http://opcfoundation.org/UA/Machinery/ProcessValues/Server/LimitsAlarmSuppression/

10.2.2 Server Facets

10.2.2.1 Overview

The following sections specify the *Facets* available for *Servers* that implement the Machinery – Process Values companion specification. Each section defines and describes a *Facet* or *Profile*.

10.2.2.2 Machinery-Process Values Base Server Facet

Table 10 defines a *Facet* that describes the base functionality to provide process values.

Table 10 – Machinery-Process Values Base Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Address Space Model	0:Address Space Base	M
Attribute Services	0:Attribute Read	M
Data Access	0:Data Access AnalogUnitRangeType	M
Data Access	0:Data Access AnalogUnitType	M
View Services	0:View Basic 2	M
View Services	0:TranslateBrowsePath	M
Machinery Process Values	Machinery Process Values Base Types	M
Machinery Process Values	Machinery Process Values Analog Object Instances	M

10.2.2.3 Machinery-Process Values Simple Device Info Server Facet

Table 11 defines a *Facet* that describes the base functionality to provide process values in the context of the device / component providing the process value. The identification of the device has to be included, optionally health information can be provided.

Table 11 – Machinery-Process Values Device Info Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	Machinery-Process Values Base Server Facet	
PA-DIM	3:PA-DIM ISignal	M
Machinery Process Values	Machinery Process Values Device Object	M
DI	2:DI DeviceHealth	O
DI	2:DI HealthDiagnosticsAlarm	O
DI	2:DI DeviceHealthProperty	O
Machinery	4:Machinery Component Identification	M

10.2.2.4 Machinery-Process Values Zero Point Adjustment Base Server Facet

Table 12 defines a *Facet* that a server can provide zero point adjustment functionality on a process value.

Table 12 – Machinery-Process Values Zero Point Adjustment Base Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	Machinery-Process Values Base Server Facet	
PA-DIM	3:PA-DIM ZeroPointAdjustment method	M
PA-DIM	3:PA-DIM Analog Signal	M
Method Services	0:Method Call	M

10.2.2.5 Machinery-Process Values Zero Point Adjustment Events Server Facet

Table 13 defines a *Facet* that a server can provide zero point adjustment on a process value including the generation of events when the adjustment is executed.

Table 13 – Machinery-Process Values Zero Point Adjustment Events Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	Machinery-Process Values Zero Point Adjustment Base Server Facet	
Profile	0:Standard Event Subscription 2022 Server Facet	
Machinery Process Values	Machinery Process Values Base EventTypes	M
Machinery Process Values	Machinery Process Values ZeroPointAdjustment Events	M

10.2.2.6 Machinery-Process Values Simulation Server Facet

Table 14 defines a *Facet* that a server can provide process values including simulation values.

Table 14 – Machinery-Process Values Simulation Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	Machinery-Process Values Base Server Facet	
PA-DIM	3:PA-DIM AnalogSignalVariable Simulation	M

10.2.2.7 Machinery-Process Values Base Process Value Setpoint Server Facet

Table 15 defines a *Facet* that a server can provide process values including a process value setpoint.

Table 15 – Machinery-Process Values Base Process Value Setpoint Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	Machinery-Process Values Base Server Facet	
Machinery Process Values	Machinery Process Values Base Process Value Setpoint	M

10.2.2.8 Machinery-Process Values Percentage Value Server Facet

Table 16 defines a *Facet* that a server can provide process values including a percentage value.

Table 16 – Machinery-Process Values Percentage Value Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	Machinery-Process Values Base Server Facet	
Machinery Process Values	Machinery Process Values Percentage Value	M

10.2.2.9 Machinery-Process Values Deviation Base Server Facet

Table 17 defines a *Facet* that a server can provide process values including deviation limits.

Table 17 – Machinery-Process Values Deviation Base Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	Machinery-Process Values Base Server Facet	
Machinery Process Values	Machinery Process Values Deviation Base	M

10.2.2.10 Machinery-Process Values Deviation AutoAdjustment Server Facet

Table 18 defines a *Facet* that a server can provide process values including at least one deviation limit with automatic adjustment. Optionally, the sensitivity of the adjustment is provided.

Table 18 – Machinery-Process Values Deviation AutoAdjustment Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	Machinery-Process Values Deviation Base Server Facet	
Machinery Process Values	Machinery Process Values Deviation AutoAdjustment	M
Machinery Process Values	Machinery Process Values Deviation Sensitivity	O

10.2.2.11 Machinery-Process Values Deviation Monitoring Server Facet

Table 19 defines a *Facet* that a server can provide process values including deviation limits and a variable to monitor if the deviation limit is reached.

Table 19 – Machinery-Process Values Deviation Monitoring Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	Machinery-Process Values Deviation Base Server Facet	
Machinery Process Values	Machinery Process Values Monitoring	M

10.2.2.12 Machinery-Process Values Deviation Alarm Server Facet

Table 20 defines a *Facet* that a server can provide process values including deviation limits and alarms if the deviation limit is reached. Optionally the alarm is represented as Object in the AddressSpace.

Table 20 – Machinery-Process Values Deviation Alarm Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	Machinery-Process Values Deviation Base Server Facet	
Profile	0:A & C Base Condition 2022 Server Facet	
Machinery Process Values	Machinery Process Values Deviation Alarm	M
Machinery Process Values	Machinery Process Values Deviation Alarm Object	O

10.2.2.13 Machinery-Process Values Deviation Alarm Suppression Server Facet

Table 21 defines a *Facet* that a server can provide process values including deviation limits and the possibility to supress alarming when the limit is reached.

Table 21 – Machinery-Process Values Deviation Alarm Suppression Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	Machinery-Process Values Deviation Base Server Facet	
Machinery Process Values	Machinery Process Values AlarmSuppression	M

10.2.2.14 Machinery-Process Values Limits Base Server Facet

Table 22 defines a *Facet* that a server can provide process values including deviation limits.

Table 22 – Machinery-Process Values Limits Base Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	Machinery-Process Values Base Server Facet	
Machinery Process Values	Machinery Process Values Limits Base	M

10.2.2.15 Machinery-Process Values Limits Monitoring Server Facet

Table 23 defines a *Facet* that a server can provide process values including absolute limits and a variable to monitor if the absolute limit is reached.

Table 23 – Machinery-Process Values Limits Monitoring Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	Machinery-Process Values Limits Base Server Facet	
Machinery Process Values	Machinery Process Values Monitoring	M

10.2.2.16 Machinery-Process Values Limits Alarm Server Facet

Table 24 defines a *Facet* that a server can provide process values including absolute limits and alarms if the absolute limit is reached. Optionally the alarm is represented as Object in the AddressSpace.

Table 24 – Machinery-Process Values Limits Alarm Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	Machinery-Process Values Limits Base Server Facet	
Profile	0:A & C Base Condition 2022 Server Facet	
Machinery Process Values	Machinery Process Values Limits Alarm	M
Machinery Process Values	Machinery Process Values Limits Alarm Object	O

10.2.2.17 Machinery-Process Values Limits Alarm Suppression Server Facet

Table 25 defines a *Facet* that a server can provide process values including absolute limits and the possibility to supress alarming when the limit is reached.

Table 25 – Machinery-Process Values Limits Alarm Suppression Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	Machinery-Process Values Limits Base Server Facet	
Machinery Process Values	Machinery Process Values AlarmSuppression	M

10.2.3 Client Facets

10.2.3.1 Overview

This specification does not define any *Facets* for *Clients*.

11 Namespaces

11.1 Namespace Metadata

Table 26 defines the namespace metadata for this document. 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 10000-5 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 10000-5.

The version information is also provided as part of the ModelTableEntry in the UANodeSet XML file. The UANodeSet XML schema is defined in OPC 10000-6.

Table 26 – NamespaceMetadata Object for this Document

Attribute	**Value**	
BrowseName	http://opcfoundation.org/UA/Machinery/ProcessValues/	
Property	**DataType**	**Value**
NamespaceUri	String	http://opcfoundation.org/UA/Machinery/ProcessValues/
NamespaceVersion	String	1.00.0
NamespacePublicationDate	DateTime	2023-05-01
IsNamespaceSubset	Boolean	False
StaticNodeIdTypes	IdType []	0
StaticNumericNodeIdRange	NumericRange []	
StaticStringNodeIdPattern	String	

Note: The *IsNamespaceSubset Property* is set to False as the UANodeSet XML file contains the complete Namespace. *Servers* only exposing a subset of the Namespace need to change the value to True.

11.2 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 27 provides a list of namespaces that may be used in a Machinery – Process Values OPC UA *Server*.

Table 27 – Namespaces used in a Machinery – Process Values Server

NamespaceURI	Description
http://opcfoundation.org/UA/	Namespace for *NodeIds* and *BrowseNames* defined in the OPC UA specification. This namespace shall have namespace index 0.
Local Server URI	Namespace for nodes defined in the local *Server*. This namespace shall have namespace index 1.
http://opcfoundation.org/UA/DI/	Namespace for *NodeIds* and *BrowseNames* defined in OPC 10000-100. The namespace index is *Server* specific.
http://opcfoundation.org/UA/Dictionary/IRDI	Namespace for *NodeIds* for IRDIs using HasDictionaryEntry (defined for example by OPC 30081). The namespace index is server specific.
http://opcfoundation.org/UA/PADIM/	Namespace for *NodeIds* and *BrowseNames* defined in OPC 30081. The namespace index is *Server* specific.
http://opcfoundation.org/UA/Machinery/	Namespace for *NodeIds* and *BrowseNames* defined in OPC 40001-1. The namespace index is *Server* specific.
http://opcfoundation.org/UA/Machinery/ProcessValues/	Namespace for *NodeIds* and *BrowseNames* defined in this document. The namespace index is *Server* specific.
Vendor specific types	A *Server* may provide vendor-specific types like types derived from *ObjectTypes* defined in this document in a vendor-specific namespace.
Vendor specific instances	A *Server* provides vendor-specific instances of the standard types or vendor-specific instances of vendor-specific types in a vendor-specific namespace. It is recommended to separate vendor specific types and vendor specific instances into two or more namespaces.

Table 28 provides a list of namespaces and their indices used for *BrowseNames* in this document. The default namespace of this document is not listed since all *BrowseNames* without prefix use this default namespace.

Table 28 – Namespaces used in this document

NamespaceURI	Namespace Index	Example
http://opcfoundation.org/UA/	0	0:EngineeringUnits
http://opcfoundation.org/UA/DI/	2	2:IDeviceHealthType
http://opcfoundation.org/UA/PADIM/	3	3:AnalogSignalType
http://opcfoundation.org/UA/Machinery/	4	4:MachineryComponentIdentificationType

Annex A
(normative)

Machinery – Process Values Namespace and mappings

A.1 NodeSet and Supplementary Files for Machinery – Process Values Information Model

The Machinery - Process Values *Information Model* is identified by the following URI:

http://opcfoundation.org/UA/Machinery/ProcessValues/

Documentation for the NamespaceUri can be found here.

The *NodeSet* associated with this version of specification can be found here:

https://reference.opcfoundation.org/nodesets/?u=http://opcfoundation.org/UA/Machinery/ProcessValues/&v=1.00.0&i=1

The *NodeSet* associated with the latest version of the specification can be found here:

https://reference.opcfoundation.org/nodesets/?u=http://opcfoundation.org/UA/Machinery/ProcessValues/&i=1

The supplementary files associated with this version of specification can be found here:

https://reference.opcfoundation.org/nodesets/?u=http://opcfoundation.org/UA/Machinery/ProcessValues/&v=1.00.0&i=2

The supplementary files associated with the latest version of the specification can be found here:

https://reference.opcfoundation.org/nodesets/?u=http://opcfoundation.org/UA/Machinery/ProcessValues/&i=2

Annex B
(informative)

Examples for Process Values

B.1 Overview

The following table provides two process values and an example of their data associated with it. Note that for the DeviationAlarm, most of the optional Nodes are not shown, and for the LimitAlarm, also most of the mandatory nodes are hidden. They are set similar to the DeviationAlarm.

Table 29 – Examples for Process Values

Node	Pressure	Temperature
SignalTag (M)	“Sigxyz123”	“T001”
AnalogSignal (M)	200	65
ValuePrecision (O)	-2 (Rounded to next 100)	-
EURange (M)	{-100.0, 250.0}	{-20.0, 180.0}
InstrumentRange (O)	{-500.0, 350.0}	{-200.0, 300.0}
EngineeringUnits (M)	Pa (Pascal)	°C
ActualValue (O)	200	65
SimulationValue (O)	100	20
SimulationState (O)	FALSE	FALSE
Damping (O)	-	-
PercentageValue (O)	60[1]	-
EURange(M)	{0.0,100.0}	-
EngineeringUnits(M)	pct (Percent)	-
LowLowLimit (O)	20	5
EngineeringUnits(M)	Pa (Pascal)	pct (Percent)
LowLimit (O)	50	10
EngineeringUnits(M)	Pa (Pascal)	pct (Percent)
HighLimit (O)	230	80
EngineeringUnits(M)	Pa (Pascal)	pct (Percent)
HighHighLimit (O)	250	90
EngineeringUnits(M)	Pa (Pascal)	pct (Percent)
Status (O)	6 (WITHIN_TOLERANCE)	7 (ABOVE_HIGH_DEVIATION)
EnumValues (M)	As in Table 3	As in Table 3
AlarmSuppression (O)	0 (No alarm suppression)	-
EnumValues (M)	As in Table 3	-
ProcessValueSetpoint (O)	200	20
.SubstituteValue (O)	210	-
LowLowDeviation (O)	-40	-
EngineeringUnits (M)	Pa (Pascal)	-
LowDeviation (O)	-20	-5
EngineeringUnits (M)	Pa (Pascal)	pct (Percent)
HighDeviation (O)	20	5
EngineeringUnits (M)	Pa (Pascal)	pct (Percent)
HighHighDeviation (O)	40	-
EngineeringUnits (M)	Pa (Pascal)	-
.DeviationSensitivity (O)	1 – Middle	-
EnumValues (M)	As in Table 7	-
.AutoDeviationAdjustment (O)	FALSE	-
.EURange (M)	{-100.0, 250.0}	{-10.0, 70.0}
.InstrumentRange (O)	{-500.0, 350.0}	-
.EngineeringUnits (M)	Pa (Pascal)	°C

DeviationAlarm (O)		
SetpointNode (M)	NodeId of Setpoint	NodeId of Setpoint
LimitState (M)		
CurrentState (M)	BAD status	“High”
CurrentState.Id (M)	BAD status	2 (StateNumber of High)
LowLowLimit (O)	-40	-
LowLimit (O)	-20	-10 (– 5 percent to °C based on EURange) [2]
HighLimit (O)	20	+10 (+5 percent to °C based on EURange) [2]
HighHighLimit (O)	40	-
ActiveState (M)	“Inactive”	“Active”
ActiveState.Id (M)	FALSE	TRUE
InputNode (M)	NodeId of SignalObject	NodeId of SignalObject
AckedState (M)	“Acknowledged”	“Acknowledged”
AckedState.Id (M)	TRUE (Auto-acknowledged)	TRUE (Auto-acknowledged)
SuppressedOrShelved (M)	FALSE	FALSE
ConditionClassId (M)	NodeId of ProcessConditionClassType (static)	NodeId of ProcessConditionClassType (static)
ConditionClassName (M)	“ProcessConditionClassType” (static)	“ProcessConditionClassType” (static)
ConditionName (M)	“DeviationAlarm” (static)	“DeviationAlarm” (static)
BranchId (M)	NULL	NULL
Retain (M)	FALSE	TRUE
EnabledState (M)	“Enabled”	“Enabled”
EnabledState.Id (M)	TRUE	TRUE
Quality (M)	GOOD	GOOD
LastSeverity (M)	600	10
Comment (M)	“” (no comment set)	“” (no comment set)
ClientUserId (M)	“” (no comment set)	“” (no comment set)
EventId (M)	123435 (Unique in Server)	123436 (Unique in Server)
EventType (M)	NodeId of ExclusiveDeviationAlarmType (Static)	NodeId of ExclusiveDeviationAlarmType (Static)
SourceNode (M)	NodeId of SignalObject	NodeId of SignalObject
SourceName (M)	Name of SignalObject	Name of SignalObject
Time (M)	2022-06-02:10:00:00	2022-06-02:10:05:00
ReceiveTime (M)	2022-06-02:10:00:00	2022-06-02:10:05:00
Message (M)	“No Deviation”	“No Deviation”
Severity (M)	10	600
LimitAlarm		
LimitState (M)		
CurrentState (M)	BAD status	BAD status
CurrentState.Id (M)	BAD status	BAD status
LowLowLimit (O)	20	-10 (5 percent to °C based on EURange) [3]
LowLimit (O)	50	0 (10 percent to °C based on EURange) [3]
HighLimit (O)	230	140 (80 percent to °C based on EURange) [3]
HighHighLimit (O)	250	160 (90 percent to °C based on EURange) [3]
ActiveState (M)	“Inactive”	“Inactive”
ActiveState.Id (M)	FALSE	FALSE

[1] The *PercentageValue* in the example is not calculated based on *EURange* but by some internal logic, also considering other parameters.

[2] The *EURange* is from -20 to 180°C, -20 is 0% and +180 is 100%. The absolute deviation value based on a percentage value is “absolute_deviation_value = percentage_value * (max_value – min_value)”. For example, for +5% it is: absolute_deviation_value = 5/100 * 180°C-(-20°C) = 5/100 * 200°C = 10°C.

[3] When calculating the absolute limits, in addition to calculating the absolute_deviation_value, it needs to be set into the context of the range, i.e. absolute_value = min_value + absolute_deviation_value. For example, for +5% it is absolute_value = -20°C +10°C = -10°C.