BS EN 62541-8:2015

BSI Standards Publication

OPC unified architecture

Part 8: Data Access

bsi.

...making excellence a habit.™

National foreword

This British Standard is the UK implementation of EN 62541-8:2015. It is identical to IEC 62541-8:2015. It supersedes BS EN 62541-8:2011 which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee AMT/7, Industrial communications: process measurement and control, including fieldbus.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

Published by BSI Standards Limited 2015

ISBN 978 0 580 83007 5
ICS 25.040.40; 25.100.01

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 May 2015.

Amendments/corrigenda issued since publication

Date	Text affected

EUROPEAN STANDARD

NORME EUROPÉENNE

EUROPÄISCHE NORM

EN 62541-8

May 2015

ICS 25.040.40; 35.100

Supersedes EN 62541-8:2011

English Version

OPC unified architecture - Part 8: Data Access (IEC 62541-8:2015)

Architecture unifiée OPC - Partie 8: Accès aux données (IEC 62541-8:2015)

OPC Unified Architecture - Teil 8: Zugriff auf Automatisierungsdaten (IEC 62541-8:2015)

European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

CEN-CENELEC Management Centre: Avenue Marnix 17, B-1000 Brussels

Foreword

The text of document 65E/381/CDV, future edition 2 of IEC 62541-8, prepared by SC 65E "Devices and integration in enterprise systems", of IEC/TC 65 "Industrial-process measurement, control and automation" was submitted to the IEC-CENELEC parallel vote and approved by CENELEC as EN 62541-8:2015.

The following dates are fixed:

- latest date by which the document has to be implemented at national level by publication of an identical national standard or by endorsement (dop) 2016-01-29
- latest date by which the national standards conflicting with the document have to be withdrawn (dow) 2018-04-29

This document supersedes EN 62541-8:2011.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CENELEC [and/or CEN] shall not be held responsible for identifying any or all such patent rights.

This document has been prepared under a mandate given to CENELEC by the European Commission and the European Free Trade Association, and supports essential requirements of EU Directive(s).

Endorsement notice

The text of the International Standard IEC 62541-8:2015 was approved by CENELEC as a European Standard without any modification.

Annex ZA
(normative)

Normative references to international publications with their corresponding European publications

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

NOTE 1 When an International Publication has been modified by common modifications, indicated by (mod), the relevant EN/HD applies.

NOTE 2 Up-to-date information on the latest versions of the European Standards listed in this annex is available here: www.cenelec.eu.

Publication	Year	Title	EN/HD	Year
IEC/TR 62541-1	-	OPC unified architecture - Part 1: Overview and concepts	CLC/TR 62541-1	-
IEC 62541-3	-	OPC unified architecture - Part 3: Address Space Model	EN 62541-3	-
IEC 62541-4	-	OPC Unified Architecture - Part 4: Services	EN 62541-4	-
IEC 62541-5	-	OPC unified architecture - Part 5: Information Model	EN 62541-5	-
UNECE 20	-	Codes for Units of Measure Used in International Trade	-	-

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

OPC UNIFIED ARCHITECTURE –

Part 8: Data Access

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 62541-8 has been prepared by subcommittee 65E: Devices and integration in enterprise systems, of IEC technical committee 65: Industrial-process measurement, control and automation.

This second edition cancels and replaces the first edition published in 2011. This edition constitutes a technical revision.

This edition includes the following significant technical changes with respect to the previous edition:

a) Clarified that deadband has to be between 0.0 and 100.0. Violations result in error Bad_DeadbandFilterInvalid (6.2)
b) Added VariableTypes handling ArrayItems and DataTypes supporting this, including complex number types. These data types are required for complex analyzer devices but seem useful for other domains as well.

The text of this standard is based on the following documents:

CDV	Report on voting
65E/381/CDV	65E/407/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

A list of all parts of the IEC 62541 series, published under the general title *OPC Unified Architecture*, can be found on the IEC website.

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

OPC UNIFIED ARCHITECTURE –

Part 8: Data Access

1 Scope

This part of IEC 62451 is part of the overall OPC Unified Architecture (OPC UA) standard series and defines the information model associated with Data Access (DA). It particularly includes additional *VariableTypes* and complementary descriptions of the *NodeClass*es and *Attributes* needed for Data Access, additional *Properties,* and other information and behaviour.

The complete address space model, including all *NodeClass*es and *Attributes* is specified in IEC 62541-3. The services to detect and access data are specified in IEC 62541-4.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC TR 62541-1, *OPC Unified Architecture - Part 1: Overview and Concepts*

IEC 62541-3, *OPC unified architecture - Part 3: Address Space Model*

IEC 62541-4, *OPC unified architecture - Part 4: Services*

IEC 62541-5, *OPC unified architecture - Part 5: Information Model*

UN/CEFACT: **UNECE Recommendation N° 20**, *Codes for Units of Measure Used in International Trade*, available at http://www.unece.org/cefact/recommendations/rec_index.htm

3 Terms, definitions and abbreviations

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC TR 62541-1, IEC 62541-3, and IEC 62541-4 as well as the following apply.

3.1.1
DataItem
link to arbitrary, live automation data, that is, data that represents currently valid information

Note 1 to entry: Examples of such data are

- device data (such as temperature sensors),
- calculated data,
- status information (open/closed, moving),
- dynamically-changing system data (such as stock quotes),
- diagnostic data.

3.1.2
AnalogItem
*DataItem*s that represent continuously-variable physical quantities (e.g., length, temperature), in contrast to the digital representation of data in discrete items

Note 1 to entry: Typical examples are the values provided by temperature sensors or pressure sensors. OPC UA defines a specific *VariableType* to identify an *AnalogItem*. *Properties* describe the possible ranges of *AnalogItem*s.

3.1.3
DiscreteItem
*DataItem*s that represent data that may take on only a certain number of possible values (e.g., OPENING, OPEN, CLOSING, CLOSED)

Note 1 to entry: Specific *VariableTypes* are used to identify *DiscreteItems* with two states or with multiple states. *Properties* specify the string values for these states.

3.1.4
ArrayItem
*DataItem*s that represent continuously-variable physical quantities and where each individual data point consists of multiple values represented by an array (e.g., the spectral response of a digital filter)

Note 1 to entry: Typical examples are the data provided by analyser devices. Specific *VariableTypes* are used to identify *ArrayItem* variants.

3.1.5
EngineeringUnits
units of measurement for *AnalogItems* that represent continuously-variable physical quantities (e.g., length, mass, time, temperature)

Note 1 to entry: This standard defines *Properties* to inform about the unit used for the *DataItem* value and about the highest and lowest value likely to be obtained in normal operation.

3.2 Abbreviations and symbols

DA Data Access
EU Engineering Unit
UA Unified Architecture

4 Concepts

Data Access deals with the representation and use of automation data in Servers.

Automation data can be located inside the *Server* or on I/O cards directly connected to the *Server*. It can also be located in sub-servers or on other devices such as controllers and input/output modules, connected by serial links via field buses or other communication links. OPC UA Data Access *Servers* provide one or more OPC UA Data Access *Clients* with transparent access to their automation data.

The links to automation data instances are called *DataItems*. Which categories of automation data are provided is completely vendor-specific. Figure 1 illustrates how the *AddressSpace* of a *Server* might consist of a broad range of different *DataItem*s.

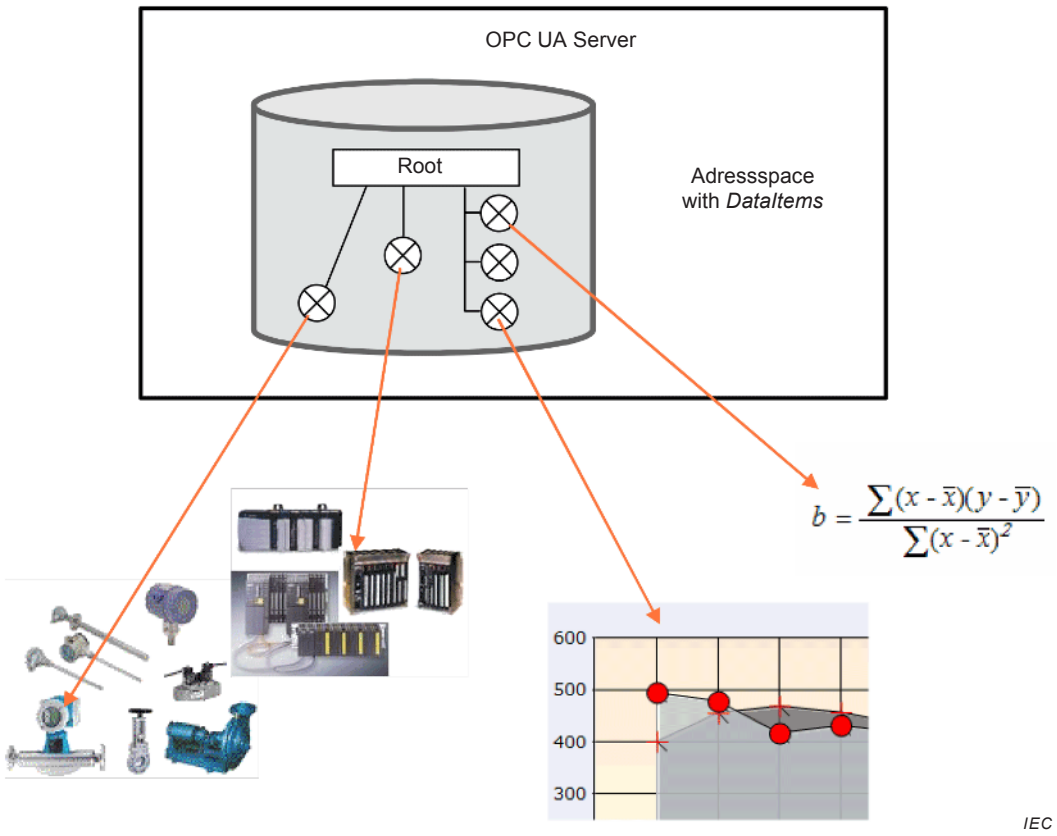

Figure 1 – OPC *DataItems* are linked to automation data

Clients may read or write *DataItem*s, or monitor them for value changes. The *Services* needed for these operations are specified in IEC 62541-4. Changes are defined as a change in status (quality) or a change in value that exceeds a client-defined range called a *Deadband*. To detect the value change, the difference between the current value and the last reported value is compared to the *Deadband*.

5 Model

5.1 General

The DataAccess model extends the variable model by defining *VariableTypes*. The *DataItemType* is the base type. *ArrayItemType*, *AnalogItemType* and *DiscreteItemType* (and its *TwoState* and *MultiState* subtypes) are specializations. See Figure 2. Each of these *VariableTypes* can be further extended to form domain or server specific *DataItems*.

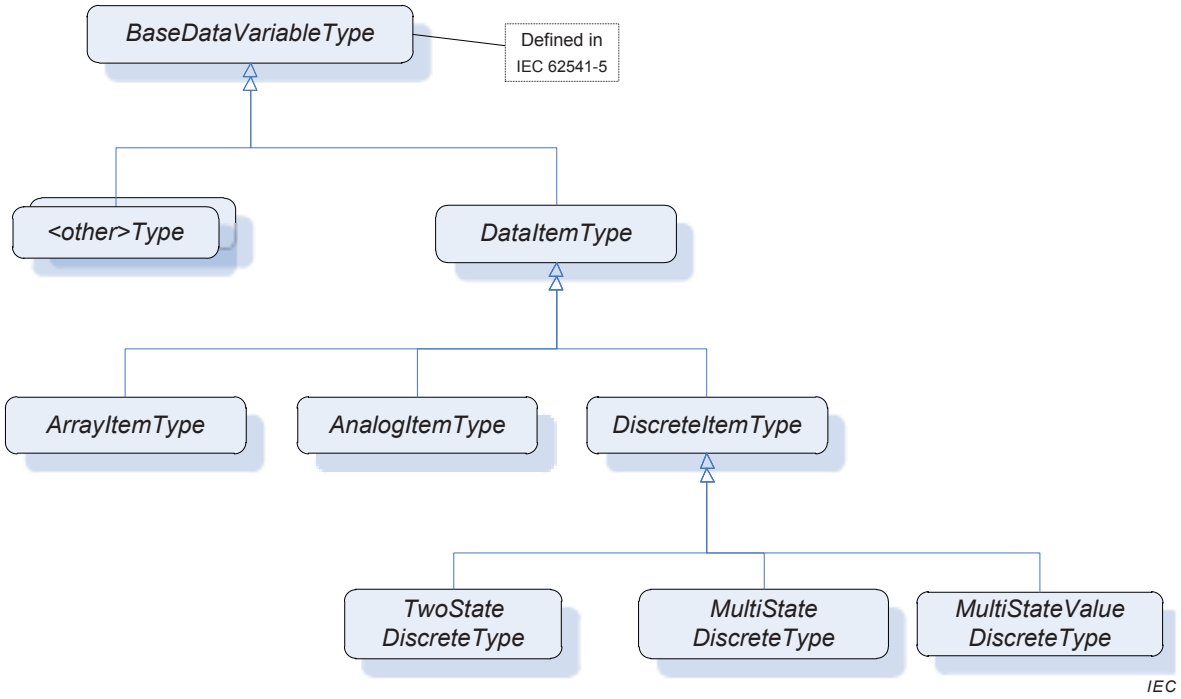

Figure 2 – *DataItem VariableType* hierarchy

5.2 SemanticsChanged

The *StatusCode* also contains an informational bit called *SemanticsChanged*.

Servers that implement Data Access shall set this Bit in notifications if certain *Properties* defined in this standard change. The corresponding *Properties* are specified individually for each *VariableType*.

Clients that use any of these *Properties* should re-read them before they process the data value.

5.3 Variable Types

5.3.1 DataItemType

This *VariableType* defines the general characteristics of a *DataItem*. All other *DataItem* Types derive from it. The *DataItemType* derives from the *BaseDataVariableType* and therefore shares the variable model as described in IEC 62541-3 and IEC 62541-5. It is formally defined in Table 1.

Table 1 – DataItemType definition

Attribute	**Value**				
BrowseName	DataItemType				
IsAbstract	False				
ValueRank	−2 (−2 = 'Any')				
DataType	BaseDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *BaseDataVariableType* defined in IEC 62541-5; i.e the *Properties* of that type are inherited.					
HasSubtype	VariableType	AnalogItemType	Defined in 5.3.2		
HasSubtype	VariableType	DiscreteItemType	Defined in 5.3.3		
HasSubtype	VariableType	ArrayItemType	Defined in 5.3.4		
HasProperty	Variable	Definition	String	PropertyType	Optional
HasProperty	Variable	ValuePrecision	Double	PropertyType	Optional

Definition is a vendor-specific, human readable string that specifies how the value of this *DataItem* is calculated. *Definition* is non-localized and will often contain an equation that can be parsed by certain clients.

Example: *Definition*::= "(TempA – 25) + TempB"

ValuePrecision specifies the maximum precision that the *Server* can maintain for the item based on restrictions in the target environment.

ValuePrecision can be used for the following *DataTypes*:

- For Float and Double values it specifies the number of digits after the decimal place.
- For DateTime values it indicates the minimum time difference in nanoseconds. For example, a ValuePrecision of 20 000 000 defines a precision of 20 ms.

The *ValuePrecision Property* is an approximation that is intended to provide guidance to a *Client*. A *Server* is expected to silently round any value with more precision that it supports. This implies that a *Client* may encounter cases where the value read back from a *Server* differs from the value that it wrote to the *Server*. This difference shall be no more than the difference suggested by this *Property*.

5.3.2 AnalogItemType

This *VariableType* defines the general characteristics of an *AnalogItem*. All other *AnalogItem* Types derive from it. The *AnalogItemType* derives from the *DataItemType*. It is formally defined in Table 2.

Table 2 – *AnalogItemType* definition

Attribute	**Value**				
BrowseName	AnalogItemType				
IsAbstract	False				
ValueRank	−2 (−2 = 'Any')				
DataType	Number				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *DataItemType* defined in 5.3.1 i.e the *Properties* of that type are inherited.					
HasProperty	Variable	InstrumentRange	Range	PropertyType	Optional
HasProperty	Variable	EURange	Range	PropertyType	Mandatory
HasProperty	Variable	EngineeringUnits	EUInformation	PropertyType	Optional

The following paragraphs describe the *Properties* of this *VariableType*. If the analog item's *Value* contains an array, the *Properties* shall apply to all elements in the array.

InstrumentRange defines the value range that can be returned by the instrument.

Example: *InstrumentRange*::= {-9999.9, 9999.9}

Although defined as optional, it is strongly recommended for *Servers* to support this *Property*. Without an *InstrumentRange* being provided, *Clients* will commonly assume the full range according to the *DataType*.

The *Range Data Type* is specified in 5.6.2.

EURange defines the value range likely to be obtained in normal operation. It is intended for such use as automatically scaling a bar graph display.

Sensor or instrument failure or deactivation can result in a returned item value which is actually outside of this range. *Client* software must be prepared to deal with this possibility. Similarly a *Client* may attempt to write a value that is outside of this range back to the server.

The exact behaviour (accept, reject, clamp, etc.) in this case is *Server*-dependent. However, in general *Servers* shall be prepared to handle this.

Example: *EURange*::= {-200.0,1400.0}

See also 6.2 for a special monitoring filter (*PercentDeadband*) which is based on the engineering unit range.

EngineeringUnits specifies the units for the *DataItem*'s value (e.g., DEGC, hertz, seconds). The *EUInformation* type is specified in 5.6.3.

Important note: Understanding the units of a measurement value is essential for a uniform system. In an open system in particular where servers from different cultures might be used, it is essential to know what the units of measurement are. Based on such knowledge, values can be converted if necessary before being used. Therefore, although defined as optional, support of the *EngineeringUnits Property* is strongly advised.

OPC UA recommends using the "**Codes for Units of Measurement"** (see UN/CEFACT: **UNECE Recommendation N° 20**). The mapping to the *EngineeringUnits Property* is specified in 5.6.3.

EXAMPLE OF UNIT MIX-UP: In 1999, the Mars Climate Orbiter crashed into the surface of Mars. The main reason was a discrepancy over the units used. The navigation software expected data in newton second; the company who built the orbiter provided data in pound-force seconds. Another, less expensive, disappointment occurs when people used to British pints order a pint in the USA, only to be served what they consider a short measure.

The *StatusCode SemanticsChanged* bit shall be set if any of the *EURange* (could change the behaviour of a *Subscription* if a *PercentDeadband* filter is used) or *EngineeringUnits* (could create problems if the client uses the value to perform calculations) *Properties* are changed (see section 5.2 for additional information).

5.3.3 DiscreteItemType

5.3.3.1 General

This VariableType is an abstract type. That is, no instances of this type can exist. However, it might be used in a filter when browsing or querying. The *DiscreteItemType* derives from the *DataItemType* and therefore shares all of its characteristics. It is formally defined in Table 3.

Table 3 – DiscreteItemType definition

Attribute	**Value**				
BrowseName	DiscreteItemType				
IsAbstract	True				
ValueRank	−2 (−2 = 'Any')				
DataType	BaseDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *DataItemType* defined in 5.2; i.e the *Properties* of that type are inherited.					
HasSubtype	VariableType	TwoStateDiscreteType	Defined in 5.3.3.2		
HasSubtype	VariableType	MultiStateDiscreteType	Defined in 5.3.3.3		
HasSubtype	VariableType	MultiStateValueDiscreteType	Defined in 5.3.3.4		

5.3.3.2 TwoStateDiscreteType

This *VariableType* defines the general characteristics of a *DiscreteItem* that can have two states. The *TwoStateDiscreteType* derives from the *DiscreteItemType*. It is formally defined in Table 4.

Table 4 – TwoStateDiscreteType definition

Attribute	Value				
BrowseName	TwoStateDiscreteType				
IsAbstract	False				
ValueRank	−2 (−2 = 'Any')				
DataType	Boolean				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *DiscreteItemType* defined in 5.3.3; i.e the *Properties* of that type are inherited.					
HasProperty	Variable	TrueState	LocalizedText	PropertyType	Mandatory
HasProperty	Variable	FalseState	LocalizedText	PropertyType	Mandatory

TrueState contains a string to be associated with this *DataItem* when it is TRUE. This is typically used for a contact when it is in the closed (non-zero) state.

for example: "RUN", "CLOSE", "ENABLE", "SAFE", etc.

FalseState contains a string to be associated with this *DataItem* when it is FALSE. This is typically used for a contact when it is in the open (zero) state.

for example: "STOP", "OPEN", "DISABLE", "UNSAFE", etc.

If the item contains an array, then the *Properties* will apply to all elements in the array.

The *StatusCode SemanticsChanged* bit shall be set if any of the *FalseState or TrueState (*changes can cause misinterpretation by users or (scripting) programs*) Properties* are changed (see section 5.2 for additional information).

5.3.3.3 MultiStateDiscreteType

This *VariableType* defines the general characteristics of a *DiscreteItem* that can have more than two states. The *MultiStateDiscreteType* derives from the *DiscreteItemType*. It is formally defined in Table 5.

Table 5 – MultiStateDiscreteType definition

Attribute	Value				
BrowseName	MultiStateDiscreteType				
IsAbstract	False				
ValueRank	−2 (−2 = 'Any')				
DataType	UInteger				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *DiscreteItemType* defined in 5.3.3; i.e the *Properties* of that type are inherited.					
HasProperty	Variable	EnumStrings	LocalizedText[]	PropertyType	Mandatory

EnumStrings is a string lookup table corresponding to sequential numeric values (0, 1, 2, etc.)

Example:

"OPEN"
"CLOSE"
"IN TRANSIT" etc.

Here the string "OPEN" corresponds to 0, "CLOSE" to 1 and "IN TRANSIT" to 2.

Clients should be prepared to handle item values outside of the range of the list; and robust servers should be prepared to handle writes of illegal values.

If the item contains an array then this lookup table shall apply to all elements in the array.

NOTE The *EnumStrings* property is also used for Enumeration *DataType*s (for the specification of this *DataType*, see IEC 62541-3).

The *StatusCode SemanticsChanged* bit shall be set if the *EnumStrings (*changes can cause misinterpretation by users or (scripting) programs*) Property* is changed (see section 5.2 for additional information).

5.3.3.4 MultiStateValueDiscreteType

This *VariableType* defines the general characteristics of a *DiscreteItem* that can have more than two states and where the state values (the enumeration) does not consist of consecutive numeric values (may have gaps) or where the enumeration is not zero-based. The *MultiStateValueDiscreteType* derives from the *DiscreteItemType*. It is formally defined in Table 6.

Table 6 – MultiStateValueDiscreteType definition

Attribute	**Value**				
BrowseName	MultiStateValueDiscreteType				
IsAbstract	False				
ValueRank	Scalar				
DataType	Number				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *DiscreteItemType* defined in 5.3.3; i.e the *Properties* of that type are inherited.					
HasProperty	Variable	EnumValues	See IEC 62541-3		Mandatory
HasProperty	Variable	ValueAsText	See IEC 62541-3		Mandatory

EnumValues is an array of *EnumValueType*. Each entry of the array represents one enumeration value with its integer notation, a human-readable representation, and help information. This represents enumerations with integers that are not zero-based or have gaps (e.g. 1, 2, 4, 8, 16). See IEC 62541-3 for the definition of this type. *MultiStateValueDiscrete Variables* expose the current integer notation in their *Value Attribute*. *Clients* will often read the *EnumValues Property* in advance and cache it to lookup a name or help whenever they receive the numeric representation.

MultiStateValueDiscrete Variables can have any numeric *Data Type*; this includes signed and unsigned integers from 8 to 64 Bit length.

The numeric representation of the current enumeration value is provided via the *Value Attribute* of the *MultiStateValueDiscrete Variable*. The *ValueAsText Property* provides the localized text representation of the enumeration value. It can be used by *Clients* only interested in displaying the text to subscribe to the *Property* instead of the *Value Attribute*.

5.3.4 ArrayItemType

5.3.4.1 General

This abstract *VariableType* defines the general characteristics of an *ArrayItem*. Values are exposed in an array but the content of the array represents a single entity like an image. Other *DataItems* might contain arrays that represent for example several values of several temperature sensors of a boiler.

ArrayItemType or its subtype shall only be used when the *Title* and *AxisScaleType Properties* can be filled with reasonable values. If this is not the case *DataItemType* and subtypes like *AnalogItemType,* which also support arrays, shall be used. The *ArrayItemType* is formally defined in Table 7.

Table 7 – ArrayItemType definition

Attribute	Value				
BrowseName	ArrayItemType				
IsAbstract	True				
ValueRank	0 (0 = OneOrMoreDimensions)				
DataType	BaseDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *DataItemType* defined in 5.3.1; i.e the *Properties* of that type are inherited.					
HasSubtype	VariableType	YArrayItemType	Defined in 5.3.4.2		
HasSubtype	VariableType	XYArrayItemType	Defined in 5.3.4.3		
HasSubtype	VariableType	ImageItemType	Defined in 5.3.4.4		
HasSubtype	VariableType	CubeItemType	Defined in 5.3.4.5		
HasSubtype	VariableType	NDimensionArrayItemType	Defined in 5.3.4.6		
HasProperty	Variable	InstrumentRange	Range	PropertyType	Optional
HasProperty	Variable	EURange	Range	PropertyType	Mandatory
HasProperty	Variable	EngineeringUnits	EUInformation	PropertyType	Mandatory
HasProperty	Variable	Title	LocalizedText	PropertyType	Mandatory
HasProperty	Variable	AxisScaleType	AxisScaleEnumeration	PropertyType	Mandatory

InstrumentRange defines the range of the *Value* of the *ArrayItem.*

EURange defines the value range of the ArrayItem likely to be obtained in normal operation. It is intended for such use as automatically scaling a bar graph display.

EngineeringUnits holds the information about the engineering units of the *Value* of the *ArrayItem*.

For additional information about *InstrumentRange*, *EURange*, and *EngineeringUnits* see the description of *AnalogItemType* in 5.3.2.

Title holds the user readable title of the *Value* of the *ArrayItem*.

AxisScaleType defines the scale to be used for the axis where the *Value* of the *ArrayItem* shall be displayed.

The *StatusCode SemanticsChanged* bit shall be set if any of the *InstrumentRange*, *EURange*, *EngineeringUnits* or *Title Properties* are changed (see 5.2 for additional information).

5.3.4.2 YArrayItemType

YArrayItemType represents a single-dimensional array of numerical values used to represent spectra or distributions where the x axis intervals are constant. *YArrayItemType* is formally defined in Table 8.

Table 8 – YArrayItemType definition

Attribute	Value				
BrowseName	YArrayItemType				
IsAbstract	False				
ValueRank	1				
DataType	BaseDataType				
ArrayDimensions	{0} (0 = UnknownSize)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *ArrayItemType* defined in 5.3.4.1					
HasProperty	Variable	XAxisDefinition	AxisInformation	PropertyType	Mandatory

The *Value* of the *YArrayItem* contains the numerical values for the Y-Axis. *Engineering Units* and *Range* for the *Value* are defined by corresponding *Properties* inherited from the *ArrayItemType*.

The *DataType* of this *VariableType* is restricted to SByte, Int16, Int32, Int64, Float, Double, *ComplexNumberType* and *DoubleComplexNumberType*.

The *XAxisDefinition Property* holds the information about the *Engineering Units* and *Range* for the X-Axis.

The *StatusCode SemanticsChanged* bit shall be set if any of the following five *Properties* are changed: *InstrumentRange, EURange, EngineeringUnits, Title* or *XAxisDefinition* (see 5.2 for additional information).

Figure 3 shows an example of how *Attributes* and *Properties* may be used in a graphical interface.

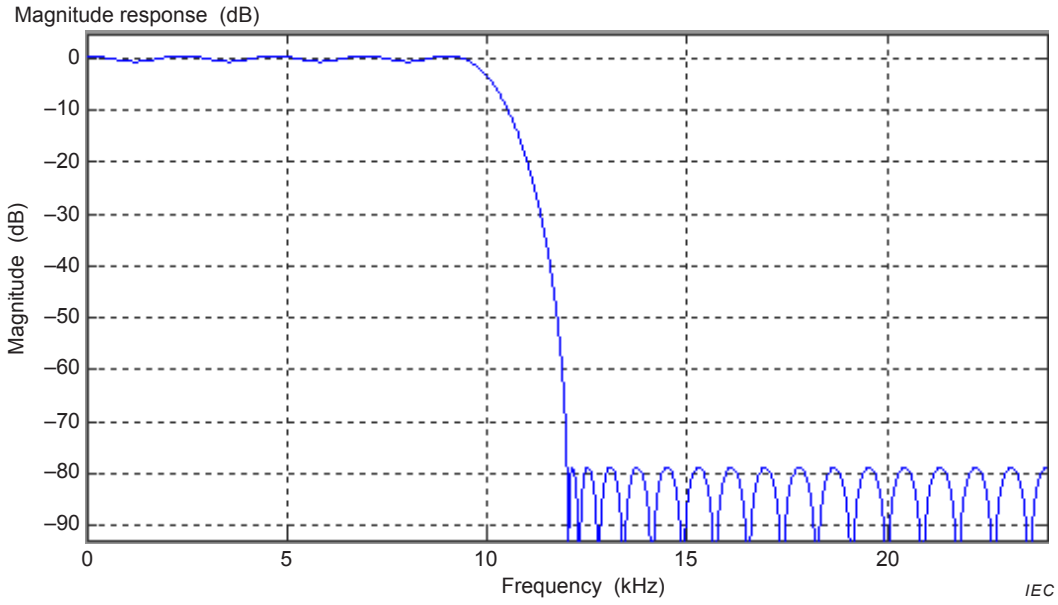

Figure 3 – Graphical view of a *YArrayItem*

Table 9 describes the values of each element presented in Figure 3.

Table 9 – *YArrayItem* item description

Attribute / Property	Item value
Description	Magnitude Response (dB)
axisScaleType	AxisScaleEnumeration.LINEAR_0
InstrumentRange.low	-90
InstrumentRange.high	5
EURange.low	-90
EURange.high	2
EngineeringUnits.namespaceUrl	http://www.opcfoundation.org/UA/units/un/cefact
EngineeringUnits.unitId	2N
EngineeringUnits.displayName	“en-us”, “dB”
EngineeringUnits.description	“en-us”, “decibel”
Title	Magnitude
XAxisDefinition.EngineeringUnits.namespaceUrl	http://www.opcfoundation.org/UA/units/un/cefact
XAxisDefinition.EngineeringUnits.unitId	kHz
XAxisDefinition.EngineeringUnits.displayName	“en-us”, “kHz”
XAxisDefinition.EngineeringUnits.description	“en-us”, kilohertz”

Attribute / Property	Item value
XAxisDefinition.Range.low	0
XAxisDefinition.Range.high	25
XAxisDefinition.title	“en-us”, “Frequency”
XAxisDefinition.axisScaleType	AxisScaleEnumeration.LINEAR_0
XAxisDefinition.axisSteps	null

Interpretation notes:

- Not all elements of this table are used in the graphic.
- The X axis is displayed in reverse order, however, the *XAxisDefinition.Range.low* shall be lower than *XAxisDefinition.Range.high*. It is only a graphical representation that reverses the display order.
- There is a constant X axis

5.3.4.3 XYArrayItemType

XYArrayItemType represents a vector of XVType values like a list of peaks, where XVType.x is the position of the peak and XVType.value is its intensity. *XYArrayItemType* is formally defined in Table 10.

Table 10 – XYArrayItemType definition

Attribute	Value				
BrowseName	XYArrayItemType				
IsAbstract	False				
ValueRank	1				
DataType	XVType (defined in 5.6.8)				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *ArrayItemType* defined in 5.3.4.1					
HasProperty	Variable	XAxisDefinition	AxisInformation	PropertyType	Mandatory

The *Value* of the X*YArrayItem* contains an array of structures (XVType) where each structure specifies the position for the X-Axis (XVType.x) and the value itself (XVType.value), used for the Y-Axis. Engineering units and range for the *Value* are defined by corresponding *Properties* inherited from the *ArrayItemType*.

XAxisDefinition Property holds the information about the *Engineering Units* and *Range* for the X-Axis.

The *axisSteps* of *XAxisDefinition* shall be set to NULL because it is not used.

The *StatusCode SemanticsChanged* bit shall be set if any of the *InstrumentRange, EURange, EngineeringUnits, Title* or *XAxisDefinition Properties* are changed (see 5.2 for additional information).

5.3.4.4 ImageItemType

ImageItemType defines the general characteristics of an ImageItem which represents a matrix of values like an image, where the pixel position is given by X which is the column and Y the row. The value is the pixel intensity.

ImageItemType is formally defined in Table 11.

Table 11 – ImageItemType definition

Attribute	Value				
BrowseName	ImageItemType				
IsAbstract	False				
ValueRank	2 (2 = two dimensional array)				
DataType	BaseDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *ArrayItemType* defined in 5.3.4.1					
HasProperty	Variable	XAxisDefinition	*AxisInformation*	PropertyType	Mandatory
HasProperty	Variable	YAxisDefinition	*AxisInformation*	PropertyType	Mandatory

Engineering units and range for the *Value* are defined by corresponding *Properties* inherited from the *ArrayItemType*.

The *DataType* of this *VariableType* is restricted to SByte, Int16, Int32, Int64, Float, Double, ComplexNumberType and DoubleComplexNumberType.

The *ArrayDimensions Attribute* for *Variables* of this type or subtypes shall use the first entry in the array ([0]) to define the number of columns and the second entry ([1]) to define the number of rows, assuming the size of the matrix is not dynamic.

XAxisDefinition Property holds the information about the engineering units and range for the X-Axis.

YAxisDefinition Property holds the information about the engineering units and range for the Y-Axis.

The *StatusCode.SemanticsChanged* bit shall be set if any of the *InstrumentRange, EURange, EngineeringUnits, Title, XAxisDefinition* or *YAxisDefinition Properties* are changed.

5.3.4.5 CubeItemType

CubeItemType represents a cube of values like a spatial particle distribution, where the particle position is given by X which is the column, Y the row and Z the depth. In the example of a spatial partical distribution, the value is the particle size. *CubeItemType* is formally defined in Table 12.

Table 12 – CubeItemType definition

Attribute	Value				
BrowseName	CubeItemType				
IsAbstract	False				
ValueRank	3 (3 = three dimensional array)				
DataType	BaseDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *ArrayItemType* defined in 5.3.4.1					
HasProperty	Variable	XAxisDefinition	*AxisInformation*	PropertyType	Mandatory
HasProperty	Variable	YAxisDefinition	*AxisInformation*	PropertyType	Mandatory
HasProperty	Variable	ZAxisDefinition	*AxisInformation*	PropertyType	Mandatory

Engineering units and range for the *Value* are defined by corresponding *Properties* inherited from the *ArrayItemType*.

The *DataType* of this *VariableType* is restricted to SByte, Int16, Int32, Int64, Float, Double, *ComplexNumberType* and *DoubleComplexNumberType*.

The *ArrayDimensions Attribute* for *Variables* of this type or subtypes should use the first entry in the array ([0]) to define the number of columns, the second entry ([1]) to define the number of rows, and the third entry ([2]) define the number of steps in the Z axis, assuming the size of the matrix is not dynamic.

XAxisDefinition Property holds the information about the engineering units and range for the X-Axis.

YAxisDefinition Property holds the information about the engineering units and range for the Y-Axis.

ZAxisDefinition Property holds the information about the engineering units and range for the Z-Axis.

The *StatusCode SemanticsChanged* bit shall be set if any of *the InstrumentRange, EURange, EngineeringUnits, Title, XAxisDefinition, YAxisDefinition* or *ZAxisDefinition Properties* are changed (see 5.2 for additional information).

5.3.4.6 NDimensionArrayItemType

This *VariableType* defines a generic multi-dimensional *ArrayItem*.

This approach minimizes the number of types however it may be proved more difficult to utilize for control system interactions.

NDimensionArrayItemType is formally defined in Table 13.

Table 13 – NDimensionArrayItemType definition

Attribute	**Value**				
BrowseName	NdimensionArrayItemType				
IsAbstract	False				
ValueRank	0 (0 = OneOrMoreDimensions)				
DataType	BaseDataType				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the *ArrayItemType* defined in 5.3.4.1					
HasProperty	Variable	AxisDefinition	AxisInformation []	PropertyType	Mandatory

The *DataType* of this *VariableType* is restricted to SByte, Int16, Int32, Int64, Float, Double, ComplexNumberType and DoubleComplexNumberType.

AxisDefinition Property holds the information about the *Engineering Units* and *Range* for all axis.

The *StatusCode SemanticsChanged* bit shall be set if any of *the InstrumentRange, EURange, EngineeringUnits, Title* or *AxisDefinition Properties* are changed (see 5.2 for additional information).

5.4 Address Space model

*DataItem*s are always defined as data components of other *Node*s in the *AddressSpace*. They are never defined by themselves. A simple example of a container for *DataItem*s would be a “Folder Object” but it can be an *Object* of any other type.

Figure 4 illustrates the basic *AddressSpace* model of a *DataItem*, in this case an *AnalogItem*.

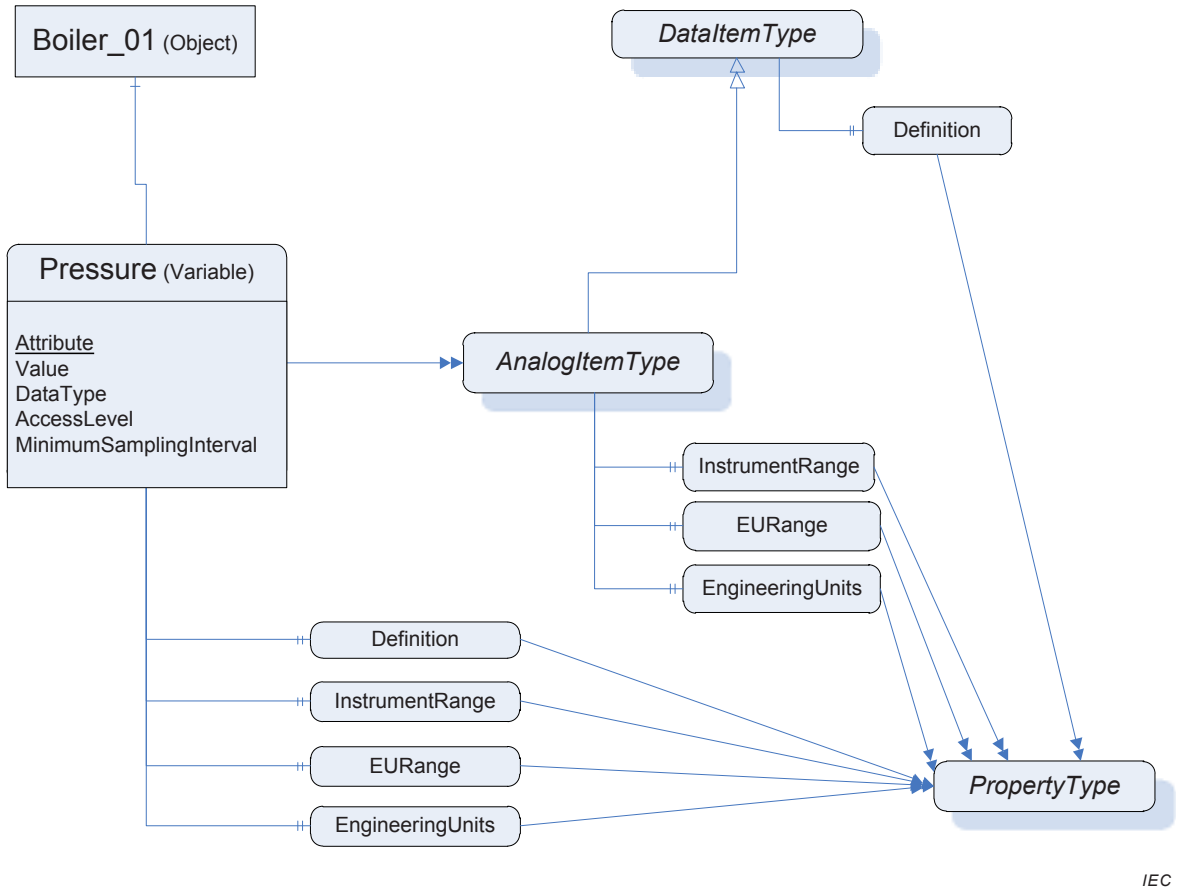

Figure 4 – Representation of DataItems in the AddressSpace

Each *DataItem* is represented by a *DataVariable* with a specific set of *Attribute*s. The *TypeDefinition* reference indicates the type of the *DataItem* (in this case the *AnalogItemType*). Additional characteristics of *DataItem*s are defined using *Properties*. The *VariableTypes* in 5.2 specify which properties may exist. These *Properties* have been found to be useful for a wide range of Data Access clients. *Servers* that want to disclose similar information should use the OPC-defined *Property* rather than one that is vendor-specific.

The above figure shows only a subset of *Attribute*s and *Properties*. Other *Attribute*s that are defined for *Variable*s in IEC 62541-3 (e.g., *Description*) may also be available.

5.5 Attributes of DataItems

This subclause lists the *Attribute*s of *Variable*s that have particular importance for Data Access. They are specified in detail in IEC 62541-3. The following *Attribute*s are particularly important for Data Access:

- Value
- DataType
- AccessLevel
- MinimumSamplingInterval

Value is the most recent value of the *Variable* that the *Server* has. Its data type is defined by the *DataType Attribute*. The *AccessLevel Attribute* defines the *Server's* basic ability to access current data and *MinimumSamplingInterval* defines how current the data is.

When a client requests the *Value Attribute* for reading or monitoring, the *Server* will always return a *StatusCode* (the quality and the *Server's* ability to access/provide the value) and,

optionally, a *ServerTimestamp* and/or a *SourceTimestamp* – based on the *Client's* request. See IEC 62541-4 for details on *StatusCode* and the meaning of the two timestamps. Specific status codes for Data Access are defined in 6.3.

5.6 DataTypes

5.6.1 Overview

Following is a description of the *DataTypes* defined in this specification.

DataTypes like *String, Boolean*, *Double* or *LocalizedText* are defined in IEC 62541-3. Their representation is specified in IEC 62541-5.

5.6.2 Range

This structure defines the *Range* for a value. Its elements are defined in Table 14.

Table 14 – *Range* DataType structure

Name	Type	Description
Range	structure	
low	Double	Lowest value in the range.
high	Double	Highest value in the range.

Its representation in the *AddressSpace* is defined in Table 15

Table 15 – *Range* definition

Attributes	Value
BrowseName	Range

5.6.3 EUInformation

This structure contains information about the *EngineeringUnits*. Its elements are defined in Table 16.

Table 16 – *EUInformation* DataType structure

Name	Type	Description
EUInformation	structure	
namespaceUri	String	Identifies the organization (company, standards organization) that defines the *EUInformation*.
unitId	Int32	Identifier for programmatic evaluation. −1 is used if a *unitId* is not available.
displayName	LocalizedText	The *displayName* of the engineering unit is typically the abbreviation of the engineering unit, for example ”h” for hour or ”m/s” for meter per second.
description	LocalizedText	Contains the full name of the engineering unit such as ”hour” or ”meter per second”.

Its representation in the *AddressSpace* is defined in Table 17

Table 17 – *EUInformation* definition

Attributes	Value
BrowseName	EUInformation

To facilitate interoperability, OPC UA specifies how to apply the widely accepted “**Codes for Units of Measurement”** published by the “United Nations Centre for Trade Facilitation and Electronic Business” (see UN/CEFACT: **UNECE Recommendation N° 20**). It uses and is based on the International System of Units (SI Units) but in addition provides a fixed code that

can be used for automated evaluation. This recommendation has been accepted by many industries on a global basis.

Table 18 contains a small excerpt of the published Annex with Code Lists:

Table 18 – Examples from the UNECE Recommendation

Excerpt from Recommendation N°. 20, Annex 1			
Common Code	**Name**	**Conversion Factor**	**Symbol**
C81	radian		rad
C25	milliradian	10^{-3} rad	mrad
MMT	millimetre	10^{-3} m	mm
HMT	hectometre	10^{2} m	hm
KTM	kilometre	10^{3} m	km
KMQ	kilogram per cubic metre	kg/m^3	kg/m^3
FAH	degree Fahrenheit	$5/9 \times K$	°F
J23	degree Fahrenheit per hour	$1{,}543\ 210 \times 10^{-4}$ K/s	°F/h

Specific columns of this table shall be used to create the *EUInformation* structure as defined by the following rules:

- The **Common Code** is represented as an alphanumeric variable length of 3 characters. It shall be used for the *EUInformation.unitId*. The following pseudo code specifies the algorithm to convert the Common Code into an Int32 as needed for *EUInformation.unitId*:

```
Int32 unitId = 0;
Int32 c;
for (i=0; i<=3;i++)
{
    c = CommonCode[i];
    if (c == 0) break;              // end of Common Code
    unitId = unitId << 8;
    unitId = unitId | c;
    }
```

- The **Symbol** field shall be copied to the *EUInformation.displayName*. The localeId field of *EUInformation.displayName* shall be empty.
- The **Name** field shall be used for *EUInformation.description*. If the name is copied, then the localeId field of *EUInformation.description* shall be empty. If the name is localized then the localeId field shall specify the correct locale.

The *EUInformation.namespaceUri* shall be http://**www.opcfoundation.org/UA/units/un/cefact**.

NOTE It will be advantageous to use Recommendation N°. 20 as specified, because it can be programmatically interpreted by generic OPC UA *Clients*. However, the *EUInformation* structure has been defined such that other standards bodies can incorporate their engineering unit definitions into OPC UA. If *Servers* use such an approach then they shall identify this standards body by using a proper *namespaceUri* in *EUInformation.namespaceUri*.

5.6.4 ComplexNumberType

This structure defines float IEEE 32 bits complex value. Its elements are defined in Table 19.

Table 19 – ComplexNumberType DataType structure

Name	Type	Description
ComplexNumberType	structure	
real	Float	Value real part
imaginary	Float	Value imaginary part

Its representation in the *AddressSpace* is defined in Table 20

Table 20 – ComplexNumberType definition

Attributes	Value
BrowseName	ComplexNumberType

5.6.5 DoubleComplexNumberType

This structure defines double IEEE 64 bits complex value. Its elements are defined in Table 21.

Table 21 – DoubleComplexNumberType DataType structure

Name	Type	Description
DoubleComplexNumberType	structure	
real	Double	Value real part
imaginary	Double	Value imaginary part

Its representation in the *AddressSpace* is defined in Table 22.

Table 22 – DoubleComplexNumberType definition

Attributes	Value
BrowseName	DoubleComplexNumberType

5.6.6 AxisInformation

This structure defines the information for auxiliary axis for *ArrayItemType Variable*s.

There are three typical uses of this structure:

a) The step between points is constant and can be predicted using the range information and the number of points. In this case, *axisSteps* can be set to NULL.

b) The step between points is not constant, but remains the same for a long period of time (from acquisition to acquisition for example). In this case, *axisSteps* contains the value of each step on the axis.

c) The step between points is not constant and changes at every update. In this case, a type like *XYArrayType* shall be used and *axisSteps* is set to NULL.

Its elements are defined in Table 23.

Table 23 – AxisInformation DataType structure

Name	Type	Description
AxisInformation	structure	
engineeringUnits	EUInformation	Holds the information about the engineering units for a given axis.
eURange	Range	Limits of the range of the axis
title	Localizedtext	User readable axis title, useful when the units are %, the Title may be “Particle size distribution”
axisScaleType	AxisScaleEnumeration	LINEAR, LOG, LN, defined by AxisSteps
axisSteps	Double[]	Specific value of each axis steps, may be set to “Null” if not used

When the steps in the axis are constant, *axisSteps* may be set to “Null” and in this case, the *Range* limits are used to compute the steps. The number of steps in the axis comes from the parent *ArrayItem.ArrayDimensions*.

5.6.7 AxisScaleEnumeration

This enumeration identifies on which type of axis the data shall be displayed. Its values are defined in Table 24.

Table 24 – AxisScaleEnumeration values

Value	Description
LINEAR_0	Linear scale
LOG_1	Log base 10 scale
LN_2	Log base e scale

Its representation in the *AddressSpace* is defined in Table 25.

Table 25 – AxisScaleEnumeration definition

Attributes	Value
BrowseName	AxisScaleEnumeration

5.6.8 XVType

This structure defines a physical value relative to a X axis and it is used as the *DataType* of the Value of *XYArrayItemType*. For details see 5.3.4.3.

Many devices can produce values that can perfectly be represented with a float IEEE 32 bits but, they can position them on the X axis with an accuracy that requires double IEEE 64 bits. For example, the peak value in an absorbance spectrum where the amplitude of the peak can be represented by a float IEEE 32 bits, but its frequency position required 10 digits which implies the use of a double IEEE 64 bits.

Its elements are defined in Table 26.

Table 26 – XVType DataType structure

Name	Type	Description
XVType	structure	
x	Double	Position on the X axis of this value
value	Float	The value itself

Its representation in the *AddressSpace* is defined in Table 27.

Table 27 – XVType definition

Attributes	Value
BrowseName	XVType

6 Data Access specific usage of Services

6.1 General

IEC 62541-4 specifies the complete set of services. The services needed for the purpose of DataAccess are:

- The *View* service set and *Query* service set to detect *DataItem*s, and their *Properties*.
- The *Attribute* service set to read or write *Attribute*s and in particular the value *Attribute*.

- The *MonitoredItem* and *Subscription* service set to set up monitoring of *DataItem*s and to receive data change notifications.

6.2 PercentDeadband

The *DataChangeFilter* in IEC 62541-4 defines the conditions under which a data change notification shall be reported. This filter contains a *deadbandValue* which can be of type *AbsoluteDeadband* or *PercentDeadband*. IEC 62541-4 already specifies the behaviour of the *AbsoluteDeadband*. This sub-clause specifies the behaviour of the *PercentDeadband* type.

DeadbandType = PercentDeadband

For this type of deadband the *deadbandValue* is defined as the percentage of the *EURange*. That is, it applies only to *AnalogItems* with an *EURange Property* that defines the typical value range for the item. This range shall be multiplied with the *deadbandValue* and then compared to the actual value change to determine the need for a data change notification. The following pseudo code shows how the deadband is calculated:

```
DataChange if (absolute value of (last cached value – current value) >
              (deadbandValue/100.0) * ((high-low) of EURange)))
```

The range of the *deadbandValue* is from 0.0 to 100.0 Percent. Specifying a *deadbandValue* outside of this range will be rejected and reported with the *StatusCode* Bad_DeadbandFilterInvalid (see Table 28).

If the Value of the *MonitoredItem* is an array, then the deadband calculation logic shall be applied to each element of the array. If an element that requires a DataChange is found, then no further deadband checking is necessary and the entire array shall be returned.

6.3 Data Access status codes

6.3.1 Overview

This subclause defines additional codes and rules that apply to the *StatusCode* when used for Data Access values.

The general structure of the *StatusCode* is specified in IEC 62541-4 and includes a set of common operational result codes that also apply to Data Access.

6.3.2 Operation level result codes

Certain conditions under which a *Variable* value was generated are only valid for automation data and in particular for device data; they are similar, but are slightly more generic than the description of data quality in the various fieldbus specifications.

In the following, Table 28 contains codes with BAD severity which indicates a failure.

Table 29 contains codes with UNCERTAIN severity which indicates that the value has been generated under sub-normal conditions.

Table 30 contains GOOD (success) codes.

Note again, that these are the codes that are specific for Data Access and supplement the codes that apply to all types of data which are defined in IEC 62541-4.

Table 28 – Operation level result codes for BAD data quality

Symbolic Id	Description
Remarks: The StatusCode Bad is defined in IEC 62541-4. It shall be used when there is no special reason why the Value is Bad.	
Bad_ConfigurationError	There is a problem with the configuration that affects the usefulness of the value.
Bad_NotConnected	The variable should receive its value from another variable, but has never been configured to do so.
Bad_DeviceFailure	There has been a failure in the device/data source that generates the value that has affected the value.
Bad_SensorFailure	There has been a failure in the sensor from which the value is derived by the device/data source. The limits bits are used to define if the limits of the value have been reached.
Remarks: • Bad_NoCommunication is defined in IEC 62541-4. It shall be used when communications to the data source is defined, but not established, and there is no last known value available.	
Bad_OutOfService	The source of the data is not operational.
~~Bad_LastKnown~~	OPC UA requires that the *Server* shall return a Null value when the *Severity* is Bad. Therefore, the Fieldbus code “Bad_LastKnown” shall be mapped to Uncertain_NoCommunicationLastUsable.
Bad_DeadbandFilterInvalid	The specified *PercentDeadband* is not between 0.0 and 100.0 or a *PercentDeadband* is not supported, since an *EURange* is not configured.
Remarks: • Bad_WaitingForInitialData is defined in IEC 62541-4.	

Table 29 – Operation level result codes for UNCERTAIN data quality

Symbolic Id	Description
Remarks: The StatusCode Uncertain is defined in IEC 62541-4. It shall be used when there is no special reason why the Value is Uncertain.	
Uncertain_ NoCommunicationLastUsable	Communication to the data source has failed. The variable value is the last value that had a good quality and it is uncertain whether this value is still current. The server timestamp in this case is the last time that the communication status was checked. The time at which the value was last verified to be true is no longer available.
Uncertain_ LastUsableValue	Whatever was updating this value has stopped doing so. This happens when an input variable is configured to receive its value from another variable and this configuration is cleared after one or more values have been received. This status/substatus is not used to indicate that a value is stale. Stale data can be detected by the client looking at the timestamps.
Uncertain_SubstituteValue	The value is an operational value that was manually overwritten.
Uncertain_InitialValue	The value is an initial value for a variable that normally receives its value from another variable. This status/substatus is set only during configuration while the variable is not operational (while it is out-of-service).
Uncertain_ SensorNotAccurate	The value is at one of the sensor limits. The Limits bits define which limit has been reached. Also set if the device can determine that the sensor has reduced accuracy (e.g. degraded analyzer), in which case the Limits bits indicate that the value is not limited.
Uncertain_ EngineeringUnitsExceeded	The value is outside of the range of values defined for this parameter. The Limits bits indicate which limit has been reached or exceeded.
Uncertain_SubNormal	The value is derived from multiple sources and has less than the required number of Good sources.

Table 30 – Operation level result codes for GOOD data quality

Symbolic Id	Description
Remarks: The StatusCode Good is defined in IEC 62541-4. It shall be used when there are no special conditions.	
Good_LocalOverride	The value has been Overridden. Typically this is means the input has been disconnected and a manually-entered value has been ”forced”.

6.3.3 LimitBits

The bottom 16 bits of the *StatusCode* are bit flags that contain additional information, but do not affect the meaning of the *StatusCode*. Of particular interest for *DataItem*s is the *LimitBits* field. In some cases, such as sensor failure it can provide useful diagnostic information.

Servers that do not support Limit have to set this field to 0.

_____

This page deliberately left blank

British Standards Institution (BSI)

BSI is the national body responsible for preparing British Standards and other standards-related publications, information and services.

BSI is incorporated by Royal Charter. British Standards and other standardization products are published by BSI Standards Limited.

About us

We bring together business, industry, government, consumers, innovators and others to shape their combined experience and expertise into standards -based solutions.

The knowledge embodied in our standards has been carefully assembled in a dependable format and refined through our open consultation process. Organizations of all sizes and across all sectors choose standards to help them achieve their goals.

Information on standards

We can provide you with the knowledge that your organization needs to succeed. Find out more about British Standards by visiting our website at bsigroup.com/standards or contacting our Customer Services team or Knowledge Centre.

Buying standards

You can buy and download PDF versions of BSI publications, including British and adopted European and international standards, through our website at bsigroup.com/shop, where hard copies can also be purchased.

If you need international and foreign standards from other Standards Development Organizations, hard copies can be ordered from our Customer Services team.

Subscriptions

Our range of subscription services are designed to make using standards easier for you. For further information on our subscription products go to bsigroup.com/subscriptions.

With **British Standards Online (BSOL)** you'll have instant access to over 55,000 British and adopted European and international standards from your desktop. It's available 24/7 and is refreshed daily so you'll always be up to date.

You can keep in touch with standards developments and receive substantial discounts on the purchase price of standards, both in single copy and subscription format, by becoming a **BSI Subscribing Member**.

PLUS is an updating service exclusive to BSI Subscribing Members. You will automatically receive the latest hard copy of your standards when they're revised or replaced.

To find out more about becoming a BSI Subscribing Member and the benefits of membership, please visit bsigroup.com/shop.

With a **Multi-User Network Licence (MUNL)** you are able to host standards publications on your intranet. Licences can cover as few or as many users as you wish. With updates supplied as soon as they're available, you can be sure your documentation is current. For further information, email bsmusales@bsigroup.com.

BSI Group Headquarters

389 Chiswick High Road London W4 4AL UK

Revisions

Our British Standards and other publications are updated by amendment or revision.

We continually improve the quality of our products and services to benefit your business. If you find an inaccuracy or ambiguity within a British Standard or other BSI publication please inform the Knowledge Centre.

Copyright

Useful Contacts:

Customer Services
Tel: +44 845 086 9001
Email (orders): orders@bsigroup.com
Email (enquiries): cservices@bsigroup.com

Subscriptions
Tel: +44 845 086 9001
Email: subscriptions@bsigroup.com

Knowledge Centre
Tel: +44 20 8996 7004
Email: knowledgecentre@bsigroup.com

Copyright & Licensing
Tel: +44 20 8996 7070
Email: copyright@bsigroup.com

bsi.

...making excellence a habit.™