



DPC: 24 / 30484420 DC

BSI Group Headquarters

389 Chiswick High Road London W4 4AL

Tel: +44 (0)20 8996 9000

Fax: +44 (0)20 8996 7400

www.bsigroup.com

Date: 29 January 2024

Origin: International

Latest date for receipt of comments: 20 March 2024

Project No. 2023/02972

Responsible committee: GEL/65/5 Devices and integration in enterprise systems

Interested committees:

Title: BS EN IEC 62541-8 OPC Unified Architecture

Part 8: Data Access

Please notify the secretary if you are aware of any keywords that might assist in classifying or identifying the standard or if the content of this standard

- i) has any issues related to 3rd party IPR, patent or copyright
- ii) affects other national standard(s)
- iii) requires additional national guidance or information

**WARNING: THIS IS A DRAFT AND MUST NOT BE REGARDED OR USED AS A BRITISH STANDARD.
THIS DRAFT IS NOT CURRENT BEYOND 20 March 2024**

This draft is issued to allow comments from interested parties; all comments will be given consideration prior to publication. No acknowledgement will normally be sent. **See overleaf for information on the submission of comments.**

No copying is allowed, in any form, without prior written permission from BSI except as permitted under the Copyright, Designs and Patent Act 1988 or for circulation within a nominating organization for briefing purposes. Electronic circulation is limited to dissemination by e-mail within such an organization by committee members.

Further copies of this draft may be purchased from BSI Shop <http://shop.bsigroup.com> or from BSI Customer Services, Tel: +44(0) 20 8996 9001 or email cservices@bsigroup.com. British, International and foreign standards are also available from BSI Customer Services.

Information on the co-operating organizations represented on the committees referenced above may be obtained from <http://standardsdevelopment.bsigroup.com>

Responsible Committee Secretary: Ms Charmaine Mukherjee (BSI)

Direct tel:

E-mail: charmaine.mukherjee@bsigroup.com

Introduction

This draft standard is based on national and international discussions. Your comments on this draft are invited and will assist in the preparation of the consequent standard.

For international standards, comments will be reviewed by the relevant UK national committee before sending the consensus UK vote and comments to the international committee, which will then decide appropriate action. If the international standard is approved, it is usual for the text to be published as a British Standard.

For national standards, comments will be reviewed by the relevant UK national committee and the resulting standards published as a British Standard.

UK Vote

Please indicate whether you consider the UK should submit a negative (with supporting technical reasons) or positive vote on this draft. Please indicate if you are aware of any reason why this draft standard should not be published as a British Standard.

Submission of Comments

- **Annotated drafts are not acceptable and will be rejected.**
- All comments should be submitted online at <https://standardsdevelopment.bsigroup.com/>. You will need to register in order to comment.

Template for comments and secretariat observations

Date: xx/xx/20xx	Document: ISO/DIS xxxx
------------------	------------------------

1	2	(3)	4	5	(6)	7
MB	Clause No./ Subclause No./Annex (e.g. 3.1)	Paragraph/Figure/ Table/Note	Type of comment	Commend (justification for change) by the MB	Proposed change by the MB	Secretariat observations on each comment submitted
	3.1	Definition 1	ed	Definition is ambiguous and needs clarifying.	Amend to read '...so that the mains connector to which no connection...'	
	6.4	Paragraph 2	te	The use of the UV photometer as an alternative cannot be supported as serious problems have been encountered in its use in the UK.	Delete reference to UV photometer.	



PROJECT NUMBER:

IEC 62541-8 ED4

DATE OF CIRCULATION:

2024-01-26

CLOSING DATE FOR VOTING:

2024-04-19

SUPERSEDES DOCUMENTS:

65E/980/RR

IEC SC 65E : DEVICES AND INTEGRATION IN ENTERPRISE SYSTEMS

SECRETARIAT:

United States of America

SECRETARY:

Mr Donald (Bob) Lattimer

OF INTEREST TO THE FOLLOWING COMMITTEES:

PROPOSED HORIZONTAL STANDARD:

☐

Other TC/SCs are requested to indicate their interest, if any, in this CDV to the secretary.

FUNCTIONS CONCERNED:

☐ EMC

☐ ENVIRONMENT

☐ QUALITY ASSURANCE

☐ SAFETY

☒ SUBMITTED FOR CENELEC PARALLEL VOTING

☐ NOT SUBMITTED FOR CENELEC PARALLEL VOTING

Attention IEC-CENELEC parallel voting

The attention of IEC National Committees, members of CENELEC, is drawn to the fact that this Committee Draft for Vote (CDV) is submitted for parallel voting.

The CENELEC members are invited to vote through the CENELEC online voting system.

This document is still under study and subject to change. It should not be used for reference purposes.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Recipients of this document are invited to submit, with their comments, notification of any relevant "In Some Countries" clauses to be included should this proposal proceed. Recipients are reminded that the CDV stage is the final stage for submitting ISC clauses. (SEE [AC/22/2007](#) OR [NEW GUIDANCE DOC](#)).

TITLE:

OPC Unified Architecture - Part 8: Data Access

PROPOSED STABILITY DATE: 2026

NOTE FROM TC/SC OFFICERS:

Copyright © 2024 International Electrotechnical Commission, IEC. All rights reserved. It is permitted to download this electronic file, to make a copy and to print out the content for the sole purpose of preparing National Committee positions. You may not copy or "mirror" the file or printed version of the document, or any part of it, for any other purpose without permission in writing from IEC.

CONTENTS

1			
2			
3	FOREWORD	vii	
4	1 Scope	1	
5	2 Normative references	1	
6	3 Terms, definitions and abbreviated terms	1	
7	3.1 Terms and definitions	1	
8	3.2 Abbreviated terms	2	
9	4 Concepts	2	
10	5 Model	4	
11	5.1 General	4	
12	5.2 SemanticsChanged	4	
13	5.3 Variable Types	4	
14	5.3.1 DataItem Type	4	
15	5.3.2 AnalogItem VariableTypes	5	
16	5.3.3 DiscreteItem Type	8	
17	5.3.4 ArrayItem Type	10	
18	5.4 Address Space model	15	
19	5.5 Attributes of DataItems	16	
20	5.6 DataTypes	17	
21	5.6.1 Overview	17	
22	5.6.2 Range	17	
23	5.6.3 EUInformation	17	
24	5.6.4 ComplexNumberType	19	
25	5.6.5 DoubleComplexNumberType	20	
26	5.6.6 AxisInformation	20	
27	5.6.7 AxisScaleEnumeration	21	
28	5.6.8 XVType	21	
29	6 Quantities and Units model	23	
30	6.1 General	23	
31	6.2 Quantities entry point	23	
32	6.3 Syntax References	23	
33	6.3.1 General	23	
34	6.3.2 Using Dictionary References	24	
35	6.3.3 Syntax Reference Identifier	25	
36	6.4 ObjectTypes	27	
37	6.4.1 QuantityType ObjectType definition	27	
38	6.4.2 UnitType and subtypes	28	
39	6.4.3 SyntaxReferenceEntryType ObjectType definition	31	
40	6.5 References	32	
41	6.5.1 HasEngineeringUnitDetails	32	
42	6.5.2 HasQuantity	32	
43	6.6 DataTypes	32	
44	6.6.1 AnnotationDataType DataType definition	32	
45	6.6.2 LinearConversionDataType DataType definition	33	
46	6.6.3 ConversionLimitEnum	34	
47	6.6.4 QuantityDimension	34	
48	7 Data Access specific usage of Services	37	

49	7.1	General.....	37
50	7.2	PercentDeadband.....	37
51	7.3	Data Access status codes.....	37
52	7.3.1	Overview.....	37
53	7.3.2	Operation level result codes	37
54	7.3.3	LimitBits	39
55		Annex A (normative) OPC COM DA to UA mapping.....	40
56	A.1	Introduction	40
57	A.2	Security Considerations	40
58	A.3	COM UA wrapper for OPC DA Server.....	40
59	A.3.1	Information Model mapping.....	40
60	A.3.2	Data and error mapping	44
61	A.3.3	Read data	47
62	A.3.4	Write Data.....	48
63	A.3.5	Subscriptions	48
64	A.4	COM UA proxy for DA Client.....	49
65	A.4.1	Guidelines.....	49
66	A.4.2	Information Model and Address Space mapping	49
67	A.4.3	Data and error mapping	53
68	A.4.4	Read data	55
69	A.4.5	Write data	56
70	A.4.6	Subscriptions	56
71		Annex B (normative) UCUM Symbols.....	58
72	B.1	Introduction - License	58
73	B.2	Representation	58
74	B.3	Tables of Terminal Symbols.....	59
75	B.1.1	Prefixes	59
76	B.1.2	Base Units	60
77	B.1.3	Derived Unit Atoms	60
78	B.1.4	Customary Unit Atoms.....	62
79	B.1.5	Other Legacy Units.....	65
80		Annex C (informative) Outline of Syntax References	69
81	C.1	UCUM Syntax Reference	69
82	C.2	QUDT Syntax Reference.....	69
83	C.3	UNECE Syntax Reference	70
84	C.4	IEC CDD Syntax Reference	70
85	C.5	LATEX_SIUNITX Syntax Reference	71
86			

FIGURES

87		
88		
89	Figure 1 – OPC <i>DataItems</i> are linked to automation data	3
90	Figure 2 – <i>DataItem VariableType</i> hierarchy	4
91	Figure 3 – Graphical view of a <i>YArrayItem</i>	12
92	Figure 4 – Representation of <i>DataItems</i> in the <i>AddressSpace</i>	16
93	Figure 5 – Enhanced <i>EUIInformation</i> example	18
94	Figure 6 – Quantity model overview	23
95	Figure 7 – References to external works	25
96	Figure 8 - <i>QuantityType</i>	27
97	Figure 9 - Units model	28
98	Figure A.1 – Sample OPC UA Information Model for OPC DA	41
99	Figure A.2 – OPC COM DA to OPC UA data and error mapping	45
100	Figure A.3 – Status Code mapping	46
101	Figure A.4 – Sample OPC DA mapping of OPC UA Information Model and Address Space ..	50
102	Figure A.5 – OPC UA to OPC DA data & error mapping	53
103	Figure A.6 – OPC UA Status Code to OPC DA quality mapping	55
104		

TABLES

105		
106		
107	Table 1 – DataItemType definition	5
108	Table 2 – BaseAnalogType definition	6
109	Table 3 – AnalogItemType definition	7
110	Table 4 – AnalogUnitType definition	7
111	Table 5 – AnalogUnitRangeType definition	8
112	Table 6 – DiscreteItemType definition	8
113	Table 7 – TwoStateDiscreteType definition	8
114	Table 8 – MultiStateDiscreteType definition	9
115	Table 9 – MultiStateValueDiscreteType definition	10
116	Table 10 – ArrayItemType definition	11
117	Table 11 – YArrayItemType definition	11
118	Table 12 – <i>YArrayItem</i> item description	12
119	Table 13 – XYArrayItemType definition	13
120	Table 14 – ImageItemType definition	14
121	Table 15 – CubeItemType definition	14
122	Table 16 – NDimensionArrayItemType definition	15
123	Table 17 – <i>Range</i> DataType structure	17
124	Table 18 – <i>Range</i> definition	17
125	Table 16 – <i>EUInformation</i> DataType structure	18
126	Table 17 – <i>EUInformation</i> definition	18
127	Table 18 – Examples from the UNECE Recommendation	19
128	Table 19 – ComplexNumberType DataType structure	19
129	Table 20 – ComplexNumberType definition	20
130	Table 21 – DoubleComplexNumberType DataType structure	20
131	Table 22 – DoubleComplexNumberType definition	20
132	Table 23 – AxisInformation DataType structure	20
133	Table 24 – AxisInformation definition	21
134	Table 25 – AxisScaleEnumeration values	21
135	Table 26 – AxisScaleEnumeration definition	21
136	Table 27 – XVType DataType structure	21
137	Table 28 – XVType definition	22
138	Table 29 – Quantities definition	23
139	Table 30 – List of Syntax References	24
140	Table 31 – Definition of NodeId for instances of the SyntaxReferenceEntryType	24
141	Table 32 – List of Syntax Reference Identifiers	25
142	Table 33 – QuantityType definition	27
143	Table 34 – QuantityType Additional Subcomponents	28
144	Table 35 – UnitType definition	29
145	Table 36 – Non-exhaustive list of well-known systems of units	29
146	Table 37 – ServerUnitType definition	29
147	Table 38 – ServerUnitType Additional Subcomponents	30

148	Table 39 – AlternativeUnitType definition	30
149	Table 40 – SyntaxReferenceEntryType Definition	31
150	Table 41 – HasEngineeringUnitDetails definition	32
151	Table 42 – HasQuantity definition	32
152	Table 43 – AnnotationDataType Structure	33
153	Table 44 – AnnotationDataType examples	33
154	Table 45 – AnnotationDataType definition	33
155	Table 46 – LinearConversionDataType Structure	33
156	Table 47 – LinearConversionDataType Definition	34
157	Table 48 – ConversionLimitEnum Items	34
158	Table 49 – ConversionLimitEnum Definition	34
159	Table 50 – QuantityDimension DataType structure	35
160	Table 51 – QuantityDimension definition	35
161	Table 52 – QuantityDimension examples	35
162	Table 53 – Operation level result codes for BAD data quality	38
163	Table 54 – Operation level result codes for UNCERTAIN data quality	38
164	Table 55 – Operation level result codes for GOOD data quality	38
165	Table A.1 – OPC COM DA to OPC UA Properties mapping	43
166	Table A.2 – DataTypes and mapping	46
167	Table A.3 – Quality mapping	47
168	Table A.4 – OPC DA Read error mapping	48
169	Table A.5 – OPC DA Write error code mapping	48
170	Table A.6 – DataTypes and Mapping	54
171	Table A.7 – Quality mapping	55
172	Table A.8 – OPC UA Read error mapping	56
173	Table A.9 – OPC UA Write error code mapping	56
174		
175		

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 fourth edition cancels and replaces the third edition published in 2020. This edition constitutes a technical revision.
- This edition includes the following significant technical changes with respect to the previous edition:
- a) Added a "Quantity Model" which can be referenced from EngineeringUnit Properties. The model defines quantities and assigned units. In addition it provides alternative units and the conversion to them.
 - b) Added additional rules for ValuePrecision Property:
 - Can also be used for other subtypes like Duration and Decimal.
 - Added rules when ValuePrecision has negative values.

226 The text of this International Standard is based on the following documents:

CDV	Report on voting
65E/XX/CDV	65E/XX/RVC

227

228 Full information on the voting for the approval of this International Standard can be found in the report
229 on voting indicated in the above table.

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

231 Throughout this document and the other parts of the IEC 62541 series, certain document conventions
232 are used:

233 *Italics* are used to denote a defined term or definition that appears in the "Terms and definition" clause
234 in one of the parts of the IEC 62541 series.

235 *Italics* are also used to denote the name of a service input or output parameter or the name of a
236 structure or element of a structure that are usually defined in tables.

237 The *italicized terms and names* are, with a few exceptions, written in camel-case (the practice of
238 writing compound words or phrases in which the elements are joined without spaces, with each
239 element's initial letter capitalized within the compound). For example, the defined term is
240 *AddressSpace* instead of Address Space. This makes it easier to understand that there is a single
241 definition for *AddressSpace*, not separate definitions for Address and Space.

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

244 The committee has decided that the contents of this document will remain unchanged until the stability
245 date indicated on the IEC website under "<http://webstore.iec.ch>" in the data related to the specific
246 document. At this date, the document will be

- 247 • reconfirmed,
248 • withdrawn,
249 • replaced by a revised edition, or
250 • amended.

251

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.

252

OPC UNIFIED ARCHITECTURE –

Part 8: Data Access

1 Scope

This part of IEC 62541 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 *NodeClasses* and *Attributes* needed for Data Access, additional *Properties*, and other information and behaviour.

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

Annex A specifies the recommended way how the information received from OPC COM Data Access (DA) Servers shall be mapped to the model in this document.

2 Normative references

The following referenced documents are indispensable for the application 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.

IEC 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*

IEC 62541-19, *OPC Unified Architecture - Part 19: Dictionary References*

UN/CEFACT: UNECE Recommendation N° 20, *Codes for Units of Measure Used in International Trade*

https://www.unece.org/cefact/codesfortrade/codes_index.html

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 62541-1, IEC 62541-3, and IEC 62541-4 and 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 that represents 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 specific *VariableTypes* to identify an *AnalogItem*. *Properties* describe the possible ranges of *AnalogItems*.

3.1.3

DiscreteItem

DataItem that represents 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 that represents 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 Abbreviated terms

DA	Data Access
EU	Engineering Unit
NaN	„Not a Number“ defined in IEEE 754
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 contain a broad range of different *DataItems*.

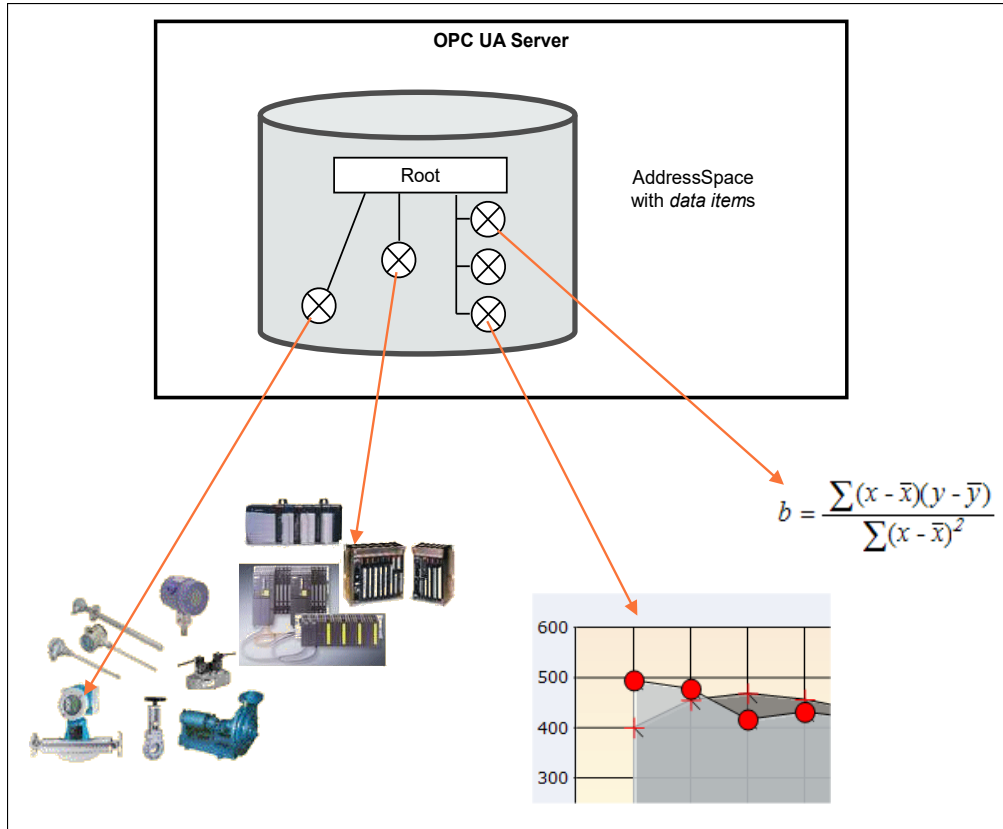


Figure 1 – OPC *DataItems* are linked to automation data

Clients may read or write *DataItems*, 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 *DataItem* type is the base type. *ArrayItem* type, *BaseAnalog* type and *DiscreteItem* type are specializations. See Figure 2. Each of these *VariableTypes* can be further extended to form domain or server specific *DataItems*.

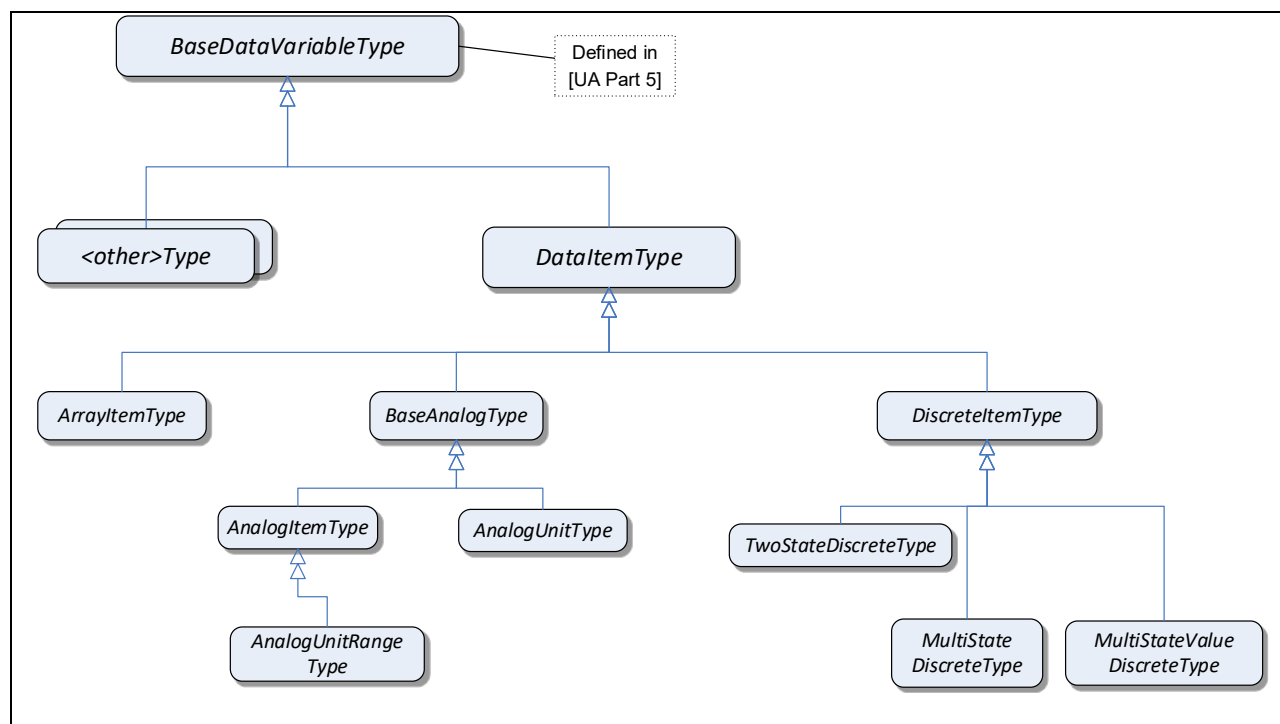


Figure 2 – *DataItem* *VariableType* hierarchy

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 *Property* values 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 DataItem

This *VariableType* defines the general characteristics of a *DataItem*. All other *DataItem* Types derive from it. The *DataItem* 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 – DatalItemType definition

Attribute	Value				
BrowseName	DatalItemType				
IsAbstract	False				
ValueRank	-2 (-2 = 'Any')				
DataType	BaseDataType				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>BaseDataVariableType</i> defined in IEC 62541-5; i.e the <i>Properties</i> of that type are inherited.					
HasSubtype	VariableType	BaseAnalogType	Defined in 5.3.2.2		
HasSubtype	VariableType	DiscreteItem	Defined in 5.3.3		
HasSubtype	VariableType	ArrayItem	Defined in 5.3.4		
HasProperty	Variable	Definition	String	PropertyType	Optional
HasProperty	Variable	ValuePrecision	Double	PropertyType	Optional
Conformance Units					
Data Access DatalItems					

Definition is a vendor-specific, human readable string that specifies how the value of this *DatalItem* 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, Double, and Decimal values it specifies the number of digits after the decimal place when it is a positive number. When it is a negative number, it specifies the number of insignificant digits to the left of the decimal place.
For example a **ValuePrecision** of -2 specifies that the precision of the *Value* is to the nearest 100. The **ValuePrecision** should always be a whole number and it shall always be interpreted as a whole number by rounding it to the nearest whole number.
- For DateTime values it shall always be a positive number which indicates the minimum time difference in nanoseconds. For example, a **ValuePrecision** of 20 000 000 defines a precision of 20 ms. The **ValuePrecision** should always be a whole number and it shall always be interpreted as a whole number by rounding it to the nearest whole number.
- ValuePrecision** can also be used for other subtypes of Double (like Duration) and other Number subtypes that can be represented by a Double.

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*.

The algorithm for rounding should follow the so-called "Banker's rounding" (aka [Round half to even](#)), in which numbers which are equidistant from the two nearest integers are rounded to the nearest even integer. Thus, 0.5 rounds down to 0; 1.5 rounds up to 2.

Other decimal fractions round as you would expect: 0.4 to 0, 0.6 to 1, 1.4 to 1, 1.6 to 2, etc. Only x.5 numbers get the "special" treatment.

5.3.2 AnalogItem VariableTypes

5.3.2.1 General

The *VariableTypes* in this subclause define the characteristics of *AnalogItems*. The types have identical semantics and *Properties* but with diverging *ModellingRules* for individual *Properties*.

The *Properties* are only described once - in 5.3.2.2. The descriptions apply to the *Properties* for the other *VariableTypes* as well.

5.3.2.2 BaseAnalogType

This *VariableType* is the base type for analog items. All *Properties* are optional. Subtypes of this base type will mandate some of the *Properties*. The *BaseAnalogType* derives from the *DataItem* type. It is formally defined in Table 2.

Table 2 – BaseAnalogType definition

Attribute	Value				
BrowseName	BaseAnalogType				
IsAbstract	False				
ValueRank	-2 (-2 = 'Any')				
DataType	Number				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>DataItem</i> type defined in 5.3.1; i.e. the <i>Properties</i> of that type are inherited.					
HasSubtype	VariableType	AnalogItem	Defined in 5.3.2.3		
HasSubtype	VariableType	AnalogUnit	Defined in 5.3.2.4		
HasProperty	Variable	InstrumentRange	Range	PropertyType	Optional
HasProperty	Variable	EURange	Range	PropertyType	Optional
HasProperty	Variable	EngineeringUnits	EUInformation	PropertyType	Optional
Conformance Units					
Data Access BaseAnalogType					

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 *InstrumentRange* *Property* may also be used to restrict a Built-in *DataType* such as *Byte* or *Int16* to a smaller range of values.

Examples:

UInt4: *InstrumentRange*::= {0, 15}
Int6: *InstrumentRange*::= {-32, 31}

The *Range* *DataType* 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 7.2 for a special monitoring filter (*PercentDeadband*) which is based on the engineering unit range.

NOTE If *EURange* is not provided on an instance, the *PercentDeadband* filter cannot be used for that instance (see clause 7.2).

EngineeringUnits specifies the units for the *DataItem*'s value (e.g., DEGC, hertz, seconds). The *EUInformation* type is specified in 5.6.3. The *NonHierarchicalReferences* *HasQuantity* (see 6.5.2) and *HasEngineeringUnitDetail* (see 6.5.1) can be used to expose further information for the unit.

It is important to note that 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.

Examples for unit mixup: 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 clause 5.2 for additional information).

5.3.2.3 AnalogItemType

This *VariableType* requires the *EURange Property*. The *AnalogItemType* derives from the *BaseAnalogType*. It is formally defined in Table 3.

Table 3 – AnalogItemType definition

Attribute	Value				
BrowseName	AnalogItemType				
IsAbstract	False				
ValueRank	-2 (-2 = 'Any')				
DataType	Number				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>BaseAnalogType</i> defined in 5.3.2.2; i.e. the <i>Properties</i> of that type are inherited.					
HasSubtype	VariableType	AnalogUnitRangeType	Defined in 5.3.2.5		
HasProperty	Variable	EURange	Range	PropertyType	Mandatory
Conformance Units					
Data Access AnalogItemType					

5.3.2.4 AnalogUnitType

This *VariableType* requires the *EngineeringUnits Property*. The *AnalogUnitType* derives from the *BaseAnalogType*. It is formally defined in Table 4.

Table 4 – AnalogUnitType definition

Attribute	Value				
BrowseName	AnalogUnitType				
IsAbstract	False				
ValueRank	-2 (-2 = 'Any')				
DataType	Number				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>BaseAnalogType</i> defined in 5.3.2.2; i.e. the <i>Properties</i> of that type are inherited.					
HasProperty	Variable	EngineeringUnits	EUInformation	PropertyType	Mandatory
Conformance Units					
Data Access AnalogUnitType					

5.3.2.5 AnalogUnitRangeType

The *AnalogUnitRangeType* derives from the *AnalogItem* and additionally requires the *EngineeringUnits* Property. It is formally defined in Table 5.

Table 5 – AnalogUnitRangeType definition

Attribute	Value				
BrowseName	AnalogUnitRangeType				
IsAbstract	False				
ValueRank	-2 (-2 = 'Any')				
DataType	Number				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AnalogItem</i> defined in 5.3.2.3; i.e. the <i>Properties</i> of that type are inherited.					
HasProperty	Variable	EngineeringUnits	EUInformation	PropertyType	Mandatory
Conformance Units					
Data Access AnalogUnitRangeType					

5.3.3 DiscreteItem

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 *DiscreteItem* derives from the *DataItem* and therefore shares all of its characteristics. It is formally defined in Table 6.

Table 6 – DiscreteItem definition

Attribute	Value				
BrowseName	DiscreteItem Type				
IsAbstract	True				
ValueRank	-2 (-2 = 'Any')				
DataType	BaseDataType				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>DataItem</i> Type defined in 5.2; i.e. the <i>Properties</i> 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		
Conformance Units					
Data Access DiscreteItem Type					

5.3.3.2 TwoStateDiscreteType

This *VariableType* defines the general characteristics of a *DiscreteItem* that can have two states. The *TwoStateDiscreteType* derives from the *DiscreteItem*. It is formally defined in Table 7.

Table 7 – TwoStateDiscreteType definition

Attribute	Value				
BrowseName	TwoStateDiscreteType				
IsAbstract	False				
ValueRank	-2 (-2 = 'Any')				
DataType	Boolean				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>DiscreteItemType</i> defined in 5.3.3; i.e. the <i>Properties</i> of that type are inherited.					
HasProperty	Variable	TrueState	LocalizedText	PropertyType	Mandatory
HasProperty	Variable	FalseState	LocalizedText	PropertyType	Mandatory
Conformance Units					
Data Access TwoState					

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 Properties* are changed (see 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 *DiscreteItem*. It is formally defined in Table 8.

Table 8 – MultiStateDiscreteType definition

Attribute	Value				
BrowseName	MultiStateDiscreteType				
IsAbstract	False				
ValueRank	-2 (-2 = 'Any')				
DataType	UInteger				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>DiscreteItem</i> defined in 5.3.3; i.e. the <i>Properties</i> of that type are inherited.					
HasProperty	Variable	EnumStrings	LocalizedText[]	PropertyType	Mandatory
Conformance Units					
Data Access MultiState					

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, by providing errorcode "Bad_OutOfRange".

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 *DataTypes* (for the specification of this *DataType*, see IEC 62541-3).

The *StatusCode SemanticsChanged* bit shall be set if the *EnumStrings Property* is changed (see 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) do not consist of consecutive numeric values (may have gaps) or where the enumeration is not zero-based. The *MultiStateValueDiscreteType* derives from the *DiscreteItem*. It is formally defined in Table 9.

Table 9 – MultiStateValueDiscreteType definition

Attribute	Value			
BrowseName	MultiStateValueDiscreteType			
IsAbstract	False			
ValueRank	-2 (-2 = 'Any')			
DataType	Number			
References	NodeClass	BrowseName	DataType TypeDefinition	ModellingRule
Subtype of the <i>DiscreteItemType</i> defined in 5.3.3; i.e. the <i>Properties</i> of that type are inherited.				
HasProperty	Variable	EnumValues	EnumValueType[] PropertyType	Mandatory
HasProperty	Variable	ValueAsText	LocalizedText PropertyType	Mandatory
Conformance Units				
Data Access MultiStateValueDiscrete				

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.

Only *DataTypes* that can be represented with *EnumValues* are allowed for *Variables* of *MultiStateValueDiscreteType*. These are Integers up to 64 Bits (signed and unsigned).

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*.

The *StatusCode SemanticsChanged* bit shall be set if the *EnumValues Property* value is changed (see clause 5.2 for additional information).

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 10.

536

Table 10 – ArrayItemType definition

Attribute	Value				
BrowseName	ArrayItemType				
IsAbstract	True				
ValueRank	0 (0 = OneOrMoreDimensions)				
DataType	BaseDataType				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>DataItemType</i> defined in 5.3.1; i.e. the <i>Properties</i> 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	NDimensionArrayItem	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
Conformance Units					
Data Access ArrayItem2Type					

537

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

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

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

543 For additional information about *InstrumentRange*, *EURange*, and *EngineeringUnits* see the
544 description of *BaseAnalogType* in 5.3.2.2.

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

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

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

550 5.3.4.2 YArrayItemType

551 *YArrayItemType* represents a single-dimensional array of numerical values used to represent
552 spectra or distributions where the x axis intervals are constant. *YArrayItemType* is formally
553 defined in Table 11.

554

Table 11 – 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 <i>ArrayItemType</i> defined in 5.3.4.1					
HasProperty	Variable	XAxisDefinition	AxisInformation	PropertyType	Mandatory
Conformance Units					
Data Access YArrayItemType					

555

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.

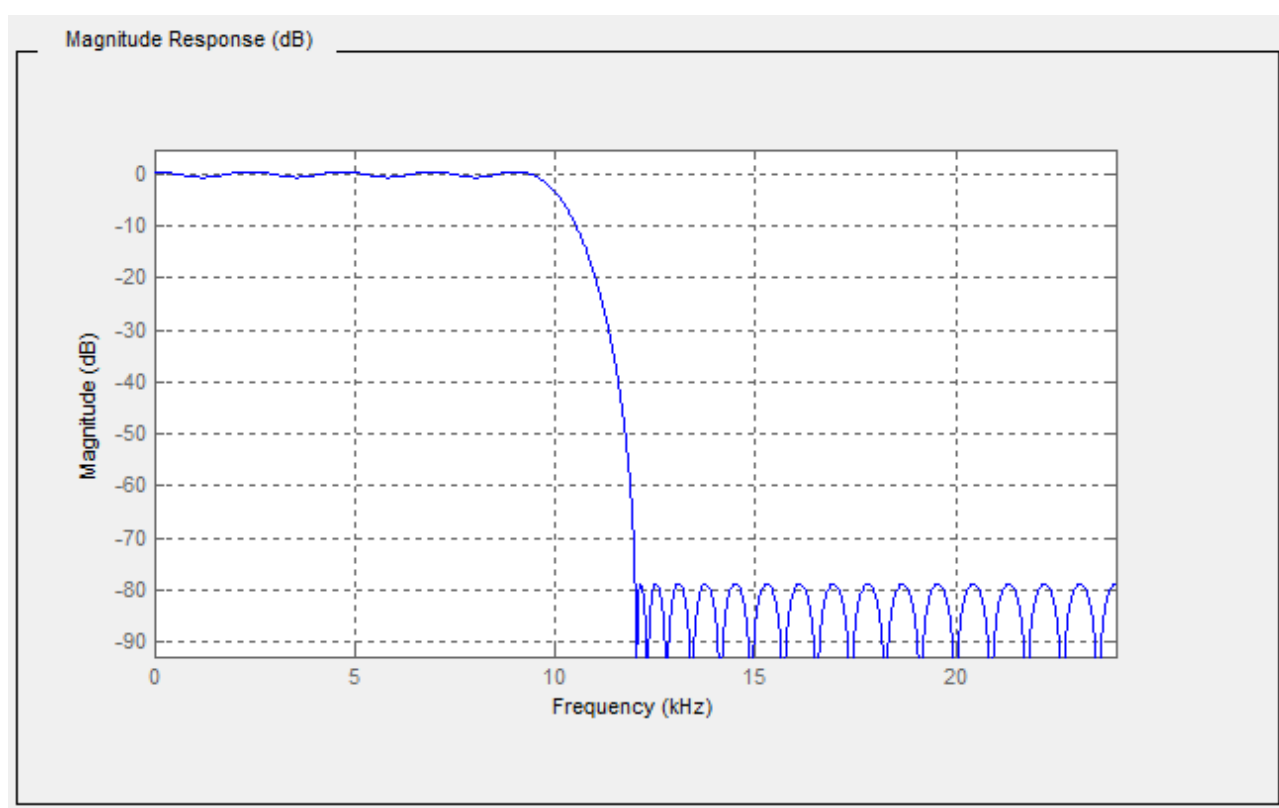


Figure 3 – Graphical view of a *YArrayItem*

Table 12 describes the values of each element presented in Figure 3.

Table 12 – *YArrayItem* item description

Attribute / Property	Item value
Description	Magnitude Response (dB)
axisScaleType	AxisScaleEnumeration.LINEAR
InstrumentRange.low	-90
InstrumentRange.high	5
EURange.low	-90
EURange.high	2
EngineeringUnits.namespaceUrl	http://www.opcfoundation.org/UA/units/un/cefact
EngineeringUnits.unitId	12878
EngineeringUnits.displayName	"en-us", "dB"

Attribute / Property	Item value
EngineeringUnits.description	"en-us", "decibel"
Title	Magnitude
XAxisDefinition.EngineeringUnits.namespaceUrl	http://www.opcfoundation.org/UA/units/un/cefact
XAxisDefinition.EngineeringUnits.unitId	4933722
XAxisDefinition.EngineeringUnits.displayName	"en-us", "kHz"
XAxisDefinition.EngineeringUnits.description	"en-us", "kilohertz"
XAxisDefinition.Range.low	0
XAxisDefinition.Range.high	25
XAxisDefinition.title	"en-us", "Frequency"
XAxisDefinition.axisScaleType	AxisScaleEnumeration.LINEAR
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 13.

Table 13 – XYArrayItemType definition

Attribute	Value				
BrowseName	XYArrayItemType				
IsAbstract	False				
ValueRank	1				
DataType	XVType				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>ArrayItemType</i> defined in 5.3.4.1					
HasProperty	Variable	XAxisDefinition	AxisInformation	PropertyType	Mandatory
Conformance Units					
Data Access XYArrayItemType					

The *Value* of the *XYArrayItem* 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 14.

Table 14 – *ImageltemType* definition

Attribute	Value				
BrowseName	ImageltemType				
IsAbstract	False				
ValueRank	2 (2 = two dimensional array)				
DataType	BaseDataType				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>ArrayItem</i> Type defined in 5.3.4.1					
HasProperty	Variable	XAxisDefinition	AxisInformation	PropertyType	Mandatory
HasProperty	Variable	YAxisDefinition	AxisInformation	PropertyType	Mandatory
Conformance Units					
Data Access ImageltemType					

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

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 CubeltemType

CubeltemType 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 particle distribution, the value is the particle size. *CubeltemType* is formally defined in Table 15.

Table 15 – *CubeltemType* definition

Attribute	Value				
BrowseName	CubeltemType				
IsAbstract	False				
ValueRank	3 (3 = three dimensional array)				
DataType	BaseDataType				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>ArrayItem</i> Type defined in 5.3.4.1					
HasProperty	Variable	XAxisDefinition	AxisInformation	PropertyType	Mandatory
HasProperty	Variable	YAxisDefinition	AxisInformation	PropertyType	Mandatory
HasProperty	Variable	ZAxisDefinition	AxisInformation	PropertyType	Mandatory
Conformance Units					
Data Access CubeltemType					

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

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 16.

Table 16 – NDimensionArrayItemType definition

Attribute	Value				
BrowseName	NDimensionArrayItemType				
IsAbstract	False				
ValueRank	0 (0 = OneOrMoreDimensions)				
DataType	BaseDataType				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>ArrayItemType</i> defined in 5.3.4.1					
HasProperty	Variable	AxisDefinition	AxisInformation []	PropertyType	Mandatory
Conformance Units					
Data Access NDimensionArrayItemType					

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

AxisDefinition Property holds the information about the *EngineeringUnits* 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

DataItems are always defined as data components of other *Nodes* in the *AddressSpace*. They are never defined by themselves. A simple example of a container for *DataItems* 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*.