



DPC: 24 / 30485900 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: 02 February 2024

Origin: International

Latest date for receipt of comments: 27 March 2024

Project No. 2024/00103

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

Interested committees:

Title: BS EN IEC 62541-5 OPC unified architecture

Part 5: Information Model

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 27 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 62541-5 ED4

DATE OF CIRCULATION:

2024-02-02

CLOSING DATE FOR VOTING:

2024-04-26

SUPERSEDES DOCUMENTS:

65E/977/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 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.	☐ NOT SUBMITTED FOR CENELEC PARALLEL VOTING

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 5: Information Model

PROPOSED STABILITY DATE: 2026

NOTE FROM TC/SC OFFICERS:

CONTENTS

FIGURES	vi
TABLES	viii
FOREWORD	xv
1 Scope	1
2 Normative references	1
3 Terms, definitions, abbreviated terms and conventions	2
3.1 Terms and definitions	2
3.2 Abbreviated terms	2
3.3 Conventions for Node descriptions	2
4 NodeIds and BrowseNames	4
4.1 NodeIds	4
4.2 BrowseNames	4
5 Common Attributes	5
5.1 General	5
5.2 Objects	5
5.3 Variables	5
5.4 VariableTypes	5
5.5 Methods	6
6 Standard ObjectTypes	6
6.1 General	6
6.2 BaseObjectType	6
6.3 ObjectTypes for the Server Object	6
6.3.1 ServerType	6
6.3.2 ServerCapabilitiesType	9
6.3.3 ServerDiagnosticsType	12
6.3.4 SessionsDiagnosticsSummaryType	13
6.3.5 SessionDiagnosticsObjectType	14
6.3.6 VendorServerInfoType	15
6.3.7 ServerRedundancyType	15
6.3.8 TransparentRedundancyType	15
6.3.9 NonTransparentRedundancyType	16
6.3.10 NonTransparentNetworkRedundancyType	16
6.3.11 OperationLimitsType	17
6.3.12 AddressSpaceFileType	18
6.3.13 NamespaceMetadataType	19
6.3.14 NamespacesType	20
6.3.15 NonTransparentBackupRedundancyType	21
6.4 ObjectTypes used as EventTypes	22
6.4.1 General	22
6.4.2 BaseEventType	22
6.4.3 AuditEventType	24
6.4.4 AuditSecurityEventType	25
6.4.5 AuditChannelEventType	26
6.4.6 AuditOpenSecureChannelEventType	26

6.4.7	AuditSessionEventType	27
6.4.8	AuditCreateSessionEventType	28
6.4.9	AuditUrlMismatchEventType	28
6.4.10	AuditActivateSessionEventType	29
6.4.11	AuditCancelEventType	29
6.4.12	AuditCertificateEventType	30
6.4.13	AuditCertificateDataMismatchEventType	30
6.4.14	AuditCertificateExpiredEventType	31
6.4.15	AuditCertificateInvalidEventType	31
6.4.16	AuditCertificateUntrustedEventType	31
6.4.17	AuditCertificateRevokedEventType	32
6.4.18	AuditCertificateMismatchEventType	32
6.4.19	AuditNodeManagementEventType	32
6.4.20	AuditAddNodesEventType	33
6.4.21	AuditDeleteNodesEventType	33
6.4.22	AuditAddReferencesEventType	34
6.4.23	AuditDeleteReferencesEventType	34
6.4.24	AuditUpdateEventType	34
6.4.25	AuditWriteUpdateEventType	35
6.4.26	AuditHistoryUpdateEventType	35
6.4.27	AuditUpdateMethodEventType	36
6.4.28	SystemEventType	36
6.4.29	DeviceFailureEventType	37
6.4.30	SystemStatusChangeEvent	37
6.4.31	BaseModelChangeEvent	37
6.4.32	GeneralModelChangeEvent	38
6.4.33	SemanticChangeEvent	38
6.4.34	EventQueueOverflowEventType	39
6.4.35	ProgressEventType	39
6.4.36	AuditClientEventType	39
6.4.37	AuditClientUpdateMethodResultEventType	40
6.5	ModellingRuleType	40
6.6	FolderType	41
6.7	DataTypeEncodingType	41
6.8	AggregateFunctionType	41
6.9	BaseInterfaceType	42
6.10	OrderedListType	42
6.11	IOrderedObjectType Definition	43
7	Standard VariableTypes	43
7.1	General	43
7.2	BaseVariableType	43
7.3	PropertyType	43
7.4	BaseDataVariableType	44
7.5	ServerVendorCapabilityType	44
7.6	ServerStatusType	44
7.7	BuildInfoType	45
7.8	ServerDiagnosticsSummaryType	45
7.9	SamplingIntervalDiagnosticsArrayType	46
7.10	SamplingIntervalDiagnosticsType	46

7.11	SubscriptionDiagnosticsArrayType	47
7.12	SubscriptionDiagnosticsType.....	47
7.13	SessionDiagnosticsArrayType	48
7.14	SessionDiagnosticsVariableType.....	49
7.15	SessionSecurityDiagnosticsArrayType	50
7.16	SessionSecurityDiagnosticsType.....	51
7.17	OptionSetType	51
7.18	SelectionListType	52
7.19	AudioVariableType	53
7.20	RationalNumberType.....	53
7.21	VectorType.....	54
7.22	3DVectorType	54
7.23	CartesianCoordinatesType	55
7.24	3DCartesianCoordinatesType.....	55
7.25	OrientationType.....	55
7.26	3DOrientationType	56
7.27	FrameType.....	56
7.28	3DFrameType	57
7.29	BitFieldType	57
8	Standard Objects and their Variables	58
8.1	General.....	58
8.2	Objects used to organise the AddressSpace structure	58
8.2.1	Overview.....	58
8.2.2	Root.....	58
8.2.3	Views	59
8.2.4	Objects	59
8.2.5	Types.....	60
8.2.6	ObjectTypes.....	60
8.2.7	VariableTypes	61
8.2.8	ReferenceTypes.....	62
8.2.9	DataTypes	63
8.2.10	EventTypes	63
8.2.11	InterfaceTypes	64
8.2.12	Locations	64
8.3	Server Object and its containing Objects	65
8.3.1	General.....	65
8.3.2	Server Object.....	66
	ModellingRule Objects	69
8.3.3	ExposesItsArray	69
8.3.4	Mandatory.....	70
8.3.5	Optional	70
8.3.6	OptionalPlaceholder	70
8.3.7	MandatoryPlaceholder	70
9	Standard Methods.....	71
9.1	GetMonitoredItems	71
9.2	ResendData	71
9.3	SetSubscriptionDurable.....	72
9.4	RequestServerStateChange	73
10	Standard Views.....	73

11	Standard ReferenceTypes.....	73
11.1	References.....	73
11.2	HierarchicalReferences	74
11.3	NonHierarchicalReferences	74
11.4	HasChild	74
11.5	Aggregates.....	75
11.6	Organizes.....	75
11.7	HasComponent	75
11.8	HasOrderedComponent.....	75
11.9	HasProperty	76
11.10	HasSubtype.....	76
11.11	HasModellingRule	76
11.12	HasTypeDefinition	77
11.13	HasEncoding	77
11.14	HasEventSource	77
11.15	HasNotifier	77
11.16	GeneratesEvent	78
11.17	AlwaysGeneratesEvent	78
11.18	HasArgumentDescription	78
11.19	HasOptionalInputArgumentDescription	79
11.20	HasInterface.....	79
11.21	HasAddIn	80
11.22	IsDeprecated.....	80
11.23	HasStructuredComponent	80
11.24	AssociatedWith	80
11.25	HasKeyValueDescription	81
12	Standard DataTypes	81
12.1	General.....	81
12.2	DataTypes defined in IEC 62541-3	81
12.2.1	BaseDataType	81
12.2.2	Boolean	82
12.2.3	ByteString	82
12.2.4	DateTime	83
12.2.5	Enumeration	84
12.2.6	Guid.....	85
12.2.7	LocalizedText.....	85
12.2.8	NodeId	85
12.2.9	Number	86
12.2.10	QualifiedName	90
12.2.11	String	90
12.2.12	Structure	92
12.2.13	XmlElement	95
12.3	DataTypes defined in IEC 62541-4	95
12.3.1	AddNodesItem	95
12.3.2	AddReferencesItem.....	95
12.3.3	ApplicationDescription	96
12.3.4	ContentFilter	96
12.3.5	DataValue	96
12.3.6	DeleteNodesItem	96

12.3.7	DeleteReferencesItem	97
12.3.8	DiagnosticInfo	97
12.3.9	ExpandedNodeId	97
12.3.10	MessageSecurityMode	97
12.3.11	NumericRange	98
12.3.12	SecurityTokenRequestType	98
12.3.13	SignedSoftwareCertificate	98
12.3.14	StatusCode	98
12.3.15	UserIdentityToken	99
12.3.16	VersionTime	100
12.4	BuildInfo	100
12.5	RedundancySupport	101
12.6	ServerState	101
12.7	RedundantServerDataType	102
12.8	SamplingIntervalDiagnosticsDataType	103
12.9	ServerDiagnosticsSummaryDataType	103
12.10	ServerStatusDataType	104
12.11	SessionDiagnosticsDataType	105
12.12	SessionSecurityDiagnosticsDataType	107
12.13	ServiceCounterDataType	107
12.14	StatusResult	108
12.15	SubscriptionDiagnosticsDataType	108
12.16	ModelChangeStructureDataType	110
12.17	SemanticChangeStructureDataType	111
12.18	BitFieldMaskDataType	111
12.19	NetworkGroupDataType	112
12.20	EndpointUrlListDataType	112
12.21	KeyValuePair	112
12.22	RationalNumber	113
12.23	Vector	113
12.24	3DVector	113
12.25	CartesianCoordinates	114
12.26	3DCartesianCoordinates	114
12.27	Orientation	115
12.28	3DOrientation	115
12.29	Frame	115
12.30	3DFrame	116
12.31	DataTypeSchemaHeader	116
12.32	DataTypeDescription	117
12.33	StructureDescription	118
12.34	EnumDescription	118
12.35	SimpleTypeDescription	118
12.36	UABinaryFileDataType	119
12.37	PortableQualifiedName	120
12.38	PortableNodeId	120
12.39	UriString	120
12.40	UnsignedRationalNumber	121
12.41	SemanticVersionString DataType	121
12.42	Handle	122

12.43	TrimmedString	123
12.44	RedundantServerMode	123
12.45	BitFieldDefinition	124
Annex A (informative)	Design decisions when modelling the server information	128
A.1	Overview	128
A.2	ServerType and Server Object	128
A.3	Typed complex Objects beneath the Server Object	128
A.4	Properties versus DataVariables	128
A.5	Complex Variables using complex DataTypes	129
A.6	Complex Variables having an array	129
A.7	Redundant information	129
A.8	Usage of the BaseDataVariableType	129
A.9	Subtyping	130
A.10	Extensibility mechanism	130
Annex B (informative)	DataTypeDictionary	131
B.1	Overview	131
Annex C (normative)	OPC Binary Type Description System	132
C.1	Concepts	132
C.2	Schema description	133
C.2.1	TypeDictionary	133
C.2.2	TypeDescription	133
C.2.3	OpaqueType	134
C.2.4	EnumeratedType	134
C.2.5	StructuredType	135
C.2.6	FieldType	135
C.2.7	EnumeratedValue	137
C.2.8	ByteOrder	137
C.2.9	ImportDirective	137
C.3	Standard Type descriptions	137
C.4	Type Description examples	138
C.5	OPC Binary XML schema	140
C.6	OPC Binary Standard TypeDictionary	141
Annex D (informative)	Geometrical references	144

FIGURES

Figure 1 – Standard AddressSpace structure	58
Figure 2 – Views organization	59
Figure 3 – Objects organization	60
Figure 4 – ObjectTypes organization	61
Figure 5 – VariableTypes organization	61
Figure 6 – ReferenceType definitions	62
Figure 7 – EventTypes organization	63
Figure 8 – InterfaceTypes Organization	64
Figure 9 – Locations organization	65
Figure 10 – Excerpt of diagnostic information of the Server	66
Figure C.1 – OPC Binary Dictionary structure	132

Figure D.1 – Simple frame chain	144
Figure D.2 – Rotations	145
Figure D.3 – Rotations of a frame.....	146

TABLES

Table 1 – Examples of DataTypes	3
Table 2 – TypeDefinition Table	4
Table 3 – Common Node Attributes	5
Table 4 – Common Object Attributes	5
Table 5 – Common Variable Attributes	5
Table 6 – Common VariableType Attributes	6
Table 7 – Common Method Attributes	6
Table 8 – BaseObjectType definition	6
Table 9 – ServerType definition	7
Table 10 – ServerCapabilitiesType definition	10
Table 11 – ServerDiagnosticsType definition	13
Table 12 – SessionsDiagnosticsSummaryType definition	14
Table 13 – SessionDiagnosticsObjectType definition	14
Table 14 – VendorServerInfoType definition	15
Table 15 – ServerRedundancyType definition	15
Table 16 – TransparentRedundancyType definition	15
Table 17 – NonTransparentRedundancyType definition	16
Table 18 – NonTransparentNetworkRedundancyType definition	16
Table 19 – OperationLimitsType definition	17
Table 20 – AddressSpaceFileType definition	18
Table 21 – NamespaceMetadataType definition	19
Table 22 – NamespacesType definition	21
Table 23 – NonTransparentBackupRedundancyType definition	21
Table 24 – BaseEventType definition	22
Table 25 – AuditEventType definition	25
Table 26 – AuditSecurityEventType definition	26
Table 27 – AuditChannelEventType definition	26
Table 28 – AuditOpenSecureChannelEventType definition	27
Table 29 – AuditSessionEventType definition	27
Table 30 – AuditCreateSessionEventType definition	28
Table 31 – AuditUrlMismatchEventType definition	29
Table 32 – AuditActivateSessionEventType definition	29
Table 33 – AuditCancelEventType definition	30
Table 34 – AuditCertificateEventType definition	30
Table 35 – AuditCertificateDataMismatchEventType definition	30
Table 36 – AuditCertificateExpiredEventType definition	31
Table 37 – AuditCertificateInvalidEventType definition	31
Table 38 – AuditCertificateUntrustedEventType definition	32
Table 39 – AuditCertificateRevokedEventType definition	32
Table 40 – AuditCertificateMismatchEventType definition	32
Table 41 – AuditNodeManagementEventType definition	33

Table 42 – AuditAddNodesEventType definition	33
Table 43 – AuditDeleteNodesEventType definition	33
Table 44 – AuditAddReferencesEventType definition	34
Table 45 – AuditDeleteReferencesEventType definition	34
Table 46 – AuditUpdateEventType definition	35
Table 47 – AuditWriteUpdateEventType definition	35
Table 48 – AuditHistoryUpdateEventType definition	36
Table 49 – AuditUpdateMethodEventType definition	36
Table 50 – SystemEventType definition	37
Table 51 – DeviceFailureEventType definition	37
Table 52 – SystemStatusChangeEvent definition	37
Table 53 – BaseModelChangeEvent definition	38
Table 54 – GeneralModelChangeEvent definition	38
Table 55 – SemanticChangeEvent definition	38
Table 56 – EventQueueOverflowEventType definition	39
Table 57 – ProgressEventType definition	39
Table 58 – AuditClientEventType definition	40
Table 59 – AuditClientUpdateMethodResultEventType definition	40
Table 60 – ModellingRuleType definition	41
Table 61 – FolderType definition	41
Table 62 – DataTypeEncodingType definition	41
Table 63 – AggregateFunctionType definition	41
Table 64 – BaseInterfaceType definition	42
Table 65 – OrderedListType Definition	42
Table 66 – OrderedListType Additional Subcomponents	42
Table 67 – IOrderedObjectType Definition	43
Table 68 – BaseVariableType definition	43
Table 69 – PropertyType definition	44
Table 70 – BaseDataVariableType definition	44
Table 71 – ServerVendorCapabilityType definition	44
Table 72 – ServerStatusType definition	45
Table 73 – BuildInfoType definition	45
Table 74 – ServerDiagnosticsSummaryType definition	46
Table 75 – SamplingIntervalDiagnosticsArrayType definition	46
Table 76 – SamplingIntervalDiagnosticsType definition	47
Table 77 – SubscriptionDiagnosticsArrayType definition	47
Table 78 – SubscriptionDiagnosticsType definition	48
Table 79 – SessionDiagnosticsArrayType definition	49
Table 80 – SessionDiagnosticsVariableType definition	49
Table 81 – SessionSecurityDiagnosticsArrayType definition	51
Table 82 – SessionSecurityDiagnosticsType definition	51
Table 83 – OptionSetType definition	52
Table 84 – SelectionListType definition	53

Table 85 – AudioVariableType definition	53
Table 86 – RationalNumberType Definition	54
Table 87 – VectorType Definition	54
Table 88 – 3DVectorType Definition	54
Table 89 – CartesianCoordinatesType Definition	55
Table 90 – 3DCartesianCoordinatesType Definition.....	55
Table 91 – OrientationType Definition	56
Table 92 – 3DOrientationType Definition	56
Table 93 – FrameType Definition.....	57
Table 94 – 3DFrameType Definition	57
Table 95 – BitFieldType definition	Error! Bookmark not defined.
Table 96 – Root definition	59
Table 97 – Views definition	59
Table 98 – Objects definition	60
Table 99 – Types definition	60
Table 100 – ObjectTypes definition	61
Table 101 – VariableTypes definition.....	62
Table 102 – ReferenceTypes definition	62
Table 103 – DataTypes definition	63
Table 104 – EventTypes definition.....	64
Table 105 – InterfaceTypes definition.....	64
Table 106 – Locations definition.....	65
Table 107 – Server definition	66
Table 108 – Server Additional Conformance Units.....	66
Table 109 – ExposesItsArray definition	70
Table 110 – Mandatory definition	70
Table 111 – Optional definition.....	70
Table 112 – OptionalPlaceholder definition	70
Table 113 – MandatoryPlaceholder definition	71
Table 114 – GetMonitoredItems Method AddressSpace definition	71
Table 115 – ResendData Method AddressSpace definition.....	72
Table 116 – SetSubscriptionDurable Method AddressSpace definition	72
Table 117 – RequestServerStateChange Method AddressSpace definition.....	73
Table 118 – References ReferenceType.....	74
Table 119 – HierarchicalReferences ReferenceType	74
Table 120 – NonHierarchicalReferences ReferenceType	74
Table 121 – HasChild ReferenceType	74
Table 122 – Aggregates ReferenceType.....	75
Table 123 – Organizes ReferenceType	75
Table 124 – HasComponent ReferenceType	75
Table 125 – HasOrderedComponent ReferenceType	76
Table 126 – HasProperty ReferenceType	76
Table 127 – HasSubtype ReferenceType.....	76

Table 128 – HasModellingRule ReferenceType	76
Table 129 – HasTypeDefinition ReferenceType	77
Table 130 – HasEncoding ReferenceType	77
Table 131 – HasEventSource ReferenceType	77
Table 132 – HasNotifier ReferenceType	78
Table 133 – GeneratesEvent ReferenceType	78
Table 134 – AlwaysGeneratesEvent ReferenceType	78
Table 135 – HasArgumentDescription ReferenceType	78
Table 136 – HasOptionalInputArgumentDescription ReferenceType	79
Table 137 – HasInterface ReferenceType	80
Table 138 – HasAddIn ReferenceType	80
Table 139 – IsDeprecated ReferenceType	80
Table 140 – HasStructuredComponent ReferenceType	80
Table 141 – AssociatedWith ReferenceType	81
Table 142 – HasKeyValueDescription ReferenceType	81
Table 143 – BaseDataType definition	81
Table 144 – Boolean definition	82
Table 145 – ByteString definition	82
Table 146 – AudioDataType definition	82
Table 147 – Image definition	82
Table 148 – ImageBMP definition	83
Table 149 – ImageGIF definition	83
Table 150 – ImageJPG definition	83
Table 151 – ImagePNG definition	83
Table 152 – DateTime definition	84
Table 153 – UtcTime definition	84
Table 154 – Enumeration definition	84
Table 155 – IdType definition	84
Table 156 – NodeClass definition	85
Table 157 – StructureType definition	85
Table 158 – Guid definition	85
Table 159 – LocalizedText definition	85
Table 160 – NodeId definition	86
Table 161 – Number definition	86
Table 162 – Decimal definition	86
Table 163 – Double definition	86
Table 164 – Duration definition	87
Table 165 – Float definition	87
Table 166 – Integer definition	87
Table 167 – Int16 definition	87
Table 168 – Int32 definition	88
Table 169 – Int64 definition	88
Table 170 – SByte definition	88

Table 171 – UInteger definition	88
Table 172 – Byte definition	89
Table 173 – UInt16 definition	89
Table 174 – AccessRestrictionType definition	89
Table 175 – UInt32 definition	89
Table 176 – UInt64 definition	90
Table 177 – QualifiedName definition	90
Table 178 – String definition	90
Table 179 – LocaleId definition	90
Table 180 – NormalizedString definition	91
Table 181 – DecimalString definition	91
Table 182 – DurationString definition	91
Table 183 – TimeString definition	91
Table 184 – DateString definition	92
Table 185 – Structure definition	92
Table 186 – Argument definition	92
Table 187 – CurrencyUnitType definition	92
Table 188 – DataTypeDefinition definition	93
Table 189 – EnumDefinition definition	93
Table 190 – StructureDefinition definition	93
Table 191 – EnumValueType definition	93
Table 192 – EnumField definition	94
Table 193 – OptionSet definition	94
Table 194 – RolePermissionType definition	94
Table 195 – StructureField definition	94
Table 196 – TimeZoneDataType definition	95
Table 197 – Union definition	95
Table 198 – XmlElement definition	95
Table 199 – AddNodesItem definition	95
Table 200 – AddReferencesItem definition	96
Table 201 – ApplicationDescription definition	96
Table 202 – ContentFilter definition	96
Table 203 – DataValue definition	96
Table 204 – DeleteNodesItem definition	97
Table 205 – DeleteReferencesItem definition	97
Table 206 – DiagnosticInfo definition	97
Table 207 – ExpandedNodeId definition	97
Table 208 – MessageSecurityMode definition	98
Table 209 – NumericRange definition	98
Table 210 – SecurityTokenRequestType definition	98
Table 211 – SignedSoftwareCertificate definition	98
Table 212 – StatusCode definition	99
Table 213 – UserIdentityToken definition	99

Table 214 – AnonymousIdentityToken definition	99
Table 215 – IssuedIdentityToken definition.....	99
Table 216 – UserNameIdentityToken definition	100
Table 217 – X509IdentityToken definition	100
Table 218 – VersionTime definition	100
Table 219 – BuildInfo Structure	100
Table 220 – BuildInfo definition	101
Table 221 – RedundancySupport values	101
Table 222 – RedundancySupport definition	101
Table 223 – ServerState values	102
Table 224 – ServerState definition	102
Table 225 – RedundantServerDataType Structure	102
Table 226 – RedundantServerDataType definition.....	103
Table 227 – SamplingIntervalDiagnosticsDataType Structure	103
Table 228 – SamplingIntervalDiagnosticsDataType definition	103
Table 229 – ServerDiagnosticsSummaryDataType Structure	104
Table 230 – ServerDiagnosticsSummaryDataType definition	104
Table 231 – ServerStatusDataType Structure.....	105
Table 232 – ServerStatusDataType definition.....	105
Table 233 – SessionDiagnosticsDataType Structure	105
Table 234 – SessionDiagnosticsDataType definition.....	107
Table 235 – SessionSecurityDiagnosticsDataType Structure	107
Table 236 – SessionSecurityDiagnosticsDataType definition	107
Table 237 – ServiceCounterDataType Structure	108
Table 238 – ServiceCounterDataType definition	108
Table 239 – StatusResult Structure	108
Table 240 – StatusResult definition	108
Table 241 – SubscriptionDiagnosticsDataType Structure.....	109
Table 242 – SubscriptionDiagnosticsDataType definition.....	110
Table 243 – ModelChangeStructureDataType Structure	110
Table 244 – ModelChangeStructureDataType definition	111
Table 245 – SemanticChangeStructureDataType Structure	111
Table 246 – SemanticChangeStructureDataType definition	111
Table 247 – BitFieldMaskDataType definition	111
Table 248 – NetworkGroupDataType Structure.....	112
Table 249 – NetworkGroupDataType definition.....	112
Table 250 – EndpointUrlListDataType Structure	112
Table 251 – EndpointUrlListDataType definition	112
Table 252 – KeyValuePair Structure.....	112
Table 253 – KeyValuePair definition.....	113
Table 254 – RationalNumber Structure.....	113
Table 256 – Vector Structure.....	113
Table 258 – 3DVector Structure	114

Table 272 – DataTypeSchemaHeader structure	117
Table 273 – DataTypeSchemaHeader definition	117
Table 274 – DataTypeDescription structure	117
Table 275 – DataTypeDescription definition	117
Table 276 – StructureDescription structure	118
Table 277 – StructureDescription definition	118
Table 278 – EnumDescription structure	118
Table 279 – EnumDescription definition	118
Table 280 – SimpleTypeDescription structure	119
Table 281 – SimpleTypeDescription definition	119
Table 282 – UABinaryFileDataType structure	119
Table 283 – UABinaryFileDataType definition	119
Table 284 – PortableQualifiedName Structure	120
Table 285 – PortableQualifiedName definition	120
Table 286 – PortableNodeId Structure	120
Table 287 – PortableNodeId definition	120
Table 288 – UriString definition	121
Table 289 – UnsignedRationalNumber structure	121
Table 290 – UnsignedRationalNumber definition	121
Table 291 – SemanticVersionString definition	121
Table 292 – Backus–Naur Form Grammar for SemanticVersionString	121
Table 293 – Handle definition	122
Table 294 – TrimmedString definition	123
Table 295 – RedundantServerMode values	123
Table 296 – RedundantServerMode definition	123
Table 297 – BitFieldDefinition Structure	Error! Bookmark not defined.
Table 298 – BitFieldDefinition definition	Error! Bookmark not defined.
Table 299 – MyBitFieldType VariableType Example	Error! Bookmark not defined.
Table 300 – MyBitFieldType BitFieldDefinitions Example	Error! Bookmark not defined.
Table 301 – MyBitFieldType SubType VariableType Example	Error! Bookmark not defined.
Table 302 – MyBitFieldType SubType BitFieldDefinitions Example ...	Error! Bookmark not defined.
Table 303 – MyBitFieldType ArrayType VariableType Example	Error! Bookmark not defined.
Table 304 – MyBitFieldType ArrayType BitFieldDefinitions Example .	Error! Bookmark not defined.
Table C.1 – TypeDictionary components	133
Table C.2 – TypeDescription components	134
Table C.3 – OpaqueType components	134
Table C.4 – EnumeratedType components	135
Table C.5 – StructuredType components	135
Table C.6 – FieldType components	135
Table C.7 – EnumeratedValue components	137
Table C.8 – ImportDirective components	137
Table C.9 – Standard Type descriptions	138

INTERNATIONAL ELECTROTECHNICAL COMMISSION

OPC UNIFIED ARCHITECTURE –

Part 5: Information Model

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 62541-5 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) Annex B has been removed and used to create Part 16.
- b) Annex C has been removed and used to create Part 20.
- c) Added Currency information model.
- d) Added information model for Interfaces and AddIns.
- e) Added information model for Method Metadata.
- f) MaxSessions, MaxSubscriptions, and MaxMonitoredItems added to capabilities.
- g) Added information model for ordered list of objects.
- h) Added PortableQualifiedName, and PortableNodeId DataTypes.
- i) Added UriString DataType.

- j) Add SemanticVersionString DataType
- k) Added AssociatedWith Reference Type
- l) Added ConfigurationVersion Property to NamespaceMetadataType.
- m) Added AuditClientEventType and AuditClientUpdateMethodResultEventType
- n) Added ModelVersion to NamespaceMetadataType
- o) Added NoTransparentBackupRedundancyType to support a Primary/Standby use case.
- p) Added BitFieldType and BitFieldDefinitionType.

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

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

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

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

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

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

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

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

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 document will remain unchanged until the stability date indicated on the IEC website under "<http://webstore.iec.ch>" in the data related to the specific document. At this date, the document 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 Specification

Part 5: Information Model

1 Scope

This part of the OPC Unified Architecture defines the Information Model. The Information Model describes standardised *Nodes* of a *Server's AddressSpace*. These *Nodes* are standardised types as well as standardised instances used for diagnostics or as entry points to server-specific *Nodes*. Thus, the Information Model defines the *AddressSpace* of an empty OPC UA *Server*. However, it is not expected that all *Servers* will provide all of these *Nodes*.

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 and errata) applies.

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

IEC 62541-3, *OPC Unified Architecture - Part 3: Address Space Model*

IEC 62541-4, *OPC Unified Architecture - Part 4: Services*

IEC 62541-6, *OPC Unified Architecture - Part 6: Mappings*

IEC 62541-7, *OPC Unified Architecture - Part 7: Profiles*

IEC 62541-8: OPC UA Specification: Part 8 – Data Access

IEC 62541-9, *OPC Unified Architecture - Part 9: Alarms and conditions*

IEC 62541-10, *OPC Unified Architecture - Part 10: Programs*

IEC 62541-11, *OPC Unified Architecture - Part 11: Historical Access*

IEC 62541-18, *OPC Unified Architecture - Part 18: Role-Based Security*

IEC 62541-20, *OPC Unified Architecture - Part 20: File Transfer*

RFC 2045, Multipurpose Internet Mail Extensions (MIME) Part One:

<http://www.ietf.org/rfc/rfc2045.txt>

RFC 2046, Multipurpose Internet Mail Extensions (MIME) Part Two:

<https://www.ietf.org/rfc/rfc2046.txt>

RFC 2047, Multipurpose Internet Mail Extensions (MIME) Part Three:

<https://www.ietf.org/rfc/rfc2047.txt>

RFC 3986, Uniform Resource Identifier (URI): Generic Syntax:

<https://www.ietf.org/rfc/rfc3986.txt>

ISO/IEC/IEEE 60559:2011, Information technology – Microprocessor Systems – Floating-Point arithmetic

<https://www.iso.org/standard/57469.html>

XML Schema Part 1, Structures

<http://www.w3.org/TR/xmlschema-1/>

XML Schema Part 2, Datatypes

<http://www.w3.org/TR/xmlschema-2/>

UTF-8, UTF-8, a transformation format of ISO 10646

<http://www.ietf.org/rfc/rfc3629.txt>

ISO/IEC 11578:1996, Information technology – Open Systems Interconnection – Remote Procedure Call (RPC) arithmetic

<https://www.iso.org/standard/2229.html>

Semantic Versioning 2.0.0

<https://semver.org/>

Unicode Database

<https://www.unicode.org/Public/UCD/latest/ucd/PropList.txt>

3 Terms, definitions, abbreviated terms and conventions

3.1 Terms and definitions

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

3.1.1

ClientUserId

String that identifies the user of the client requesting an action

Note 1 to entry: The *ClientUserId* is obtained directly or indirectly from the *UserIdentityToken* passed by the *Client* in the *ActivateSession Service* call or from the *authenticationToken* in the *requestHeader* of a *SessionlessInvoke Service* call. See 6.4.3 for details.

3.2 Abbreviated terms

UA Unified Architecture

XML Extensible Markup Language

3.3 Conventions for Node descriptions

Node definitions are specified using tables (see Table 2).

Attributes are defined by providing the *Attribute* name and a value, or a description of the value.

References are defined by providing the *ReferenceType* name, the *BrowseName* of the *TargetNode* and its *NodeClass*.

- If the *TargetNode* is a component of the *Node* being defined in the table, the *Attributes* of the composed *Node* are defined in the same row of the table.
- The *DataType* is only specified for *Variables*; "[<number>]" indicates a single-dimensional array, for multi-dimensional arrays the expression is repeated for each dimension (e.g. [2][3] for a two-dimensional array). For all arrays the *ArrayDimensions* is set as identified by <number> values. If no <number> is set, the corresponding dimension is set to 0, indicating an unknown size. If no number is provided at all the *ArrayDimensions* can be omitted. If no brackets are provided, it identifies a scalar *DataType* and the *ValueRank* is set to the corresponding value (see IEC 62541-3). In addition, *ArrayDimensions* is set to null or is omitted. If it can be Any or ScalarOrOneDimension, the value is put into "{<value>}", so either "{Any}" or "{ScalarOrOneDimension}" and the *ValueRank* is set to the corresponding value (see

IEC 62541-3) and the *ArrayDimensions* is set to null or is omitted. Examples are given in Table 1.

Table 1 – Examples of DataTypes

Notation	Data-Type	Value-Rank	Array-Dimensions	Description
Int32	Int32	-1	omitted or null	A scalar Int32.
Int32[]	Int32	1	omitted or {0}	Single-dimensional array of Int32 with an unknown size.
Int32[][]	Int32	2	omitted or {0,0}	Two-dimensional array of Int32 with unknown sizes for both dimensions.
Int32[3][]	Int32	2	{3,0}	Two-dimensional array of Int32 with a size of 3 for the first dimension and an unknown size for the second dimension.
Int32[5][3]	Int32	2	{5,3}	Two-dimensional array of Int32 with a size of 5 for the first dimension and a size of 3 for the second dimension.
Int32{Any}	Int32	-2	omitted or null	An Int32 where it is unknown if it is scalar or array with any number of dimensions.
Int32{ScalarOrOneDimension}	Int32	-3	omitted or null	An Int32 where it is either a single-dimensional array or a scalar.

- The *TypeDefinition* is specified for *Objects* and *Variables*.
- The *TypeDefinition* column specifies a symbolic name for a *NodeId*, i.e., the specified *Node* points with a *HasTypeDefinition Reference* to the corresponding *Node*.
- The *ModellingRule* of the referenced component is provided by specifying the symbolic name of the rule in the *ModellingRule* column. In the *AddressSpace*, the *Node* shall use a *HasModellingRule Reference* to point to the corresponding *ModellingRule Object*.

If the *NodeId* of a *DataType* is provided, the symbolic name of the *Node* representing the *DataType* shall be used.

Nodes of all other *NodeClasses* cannot be defined in the same table; therefore only the used *ReferenceType*, their *NodeClass* and their *BrowseName* are specified. A reference to another part of this document points to their definition.

Table 2 illustrates the table. If no components are provided, the *DataType*, *TypeDefinition* and *ModellingRule* columns may be omitted and only a *Comment* column is introduced to point to the *Node* definition.

Each *Node* defined in this specification has *ConformanceUnits* defined in IEC 62541-7 that require the *Node* to be in the *AddressSpace*. If a *Server* supports a *ConformanceUnit*, it shall expose the *Nodes* related to the *ConformanceUnit* in its *AddressSpace*. If two *Nodes* are exposed, all *References* between the *Nodes* defined in this specification shall be exposed as well.

The relations between *Nodes* and *ConformanceUnits* are defined at the end of the tables defining *Nodes*, one row per *ConformanceUnit*. The *ConformanceUnit* is reflected with a *Category* element in the *UANodeSet* file (see IEC 62541-6).

The *Nodes* defined in a table are not only the *Node* defined on top level, for example an *ObjectType*, but also the *Nodes* that are referenced, as long as they are not defined in other tables. For example, the *ObjectType ServerType* defines its *InstanceDeclarations* in the same table, so the *InstanceDeclarations* are also bound to the *ConformanceUnits* defined for the table. The table even indirectly defines additional *InstanceDeclarations* as components of the top-level *InstanceDeclarations*, that are not directly visible in the table. The *TypeDefinitions* and *DataTypes* used in the *InstanceDeclarations*, and the *ReferenceTypes* are defined in their individual tables and not in the table itself, therefore they are not bound to the *ConformanceUnits* of the table. As another example, the *OrderedListType* references the *GeneralModelChangeEvent*, which is defined in another table, and therefore not bound to

the *ConformanceUnits* of the table. In case of ambiguity, the Category element of the *UaNodeSet* file provides the precise information.

Table 2 – TypeDefinition Table

Attribute	Value				
Attribute name	Attribute value. If it is an optional Attribute that is not set "--" will be used.				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
ReferenceType name	NodeClass of the TargetNode.	BrowseName of the target Node.	DataType of the referenced Node, only applicable for Variables.	TypeDefinition of the referenced Node, only applicable for Variables and Objects.	Referenced ModellingRule of the referenced Object.
NOTE Notes referencing footnotes of the table content.					
Conformance Units					
Name of conformance unit, one row per conformance unit					

Components of *Nodes* can be complex that is containing components by themselves. The *TypeDefinition*, *NodeClass*, *DataType* and *ModellingRule* can be derived from the type definitions, and the symbolic name can be created as defined in 4.1. Therefore those containing components are not explicitly specified; they are implicitly specified by the type definitions.

4 NodeIds and BrowseNames

4.1 NodeIds

The *NodeIds* of all *Nodes* described in this standard are only symbolic names. IEC 62541-6 defines the actual *NodeIds*.

The symbolic name of each *Node* defined in this standard is its *BrowseName*, or, when it is part of another *Node*, the *BrowseName* of the other *Node*, a ".", and the *BrowseName* of itself. In this case "part of" means that the whole has a *HasProperty* or *HasComponent Reference* to its part. Since all *Nodes* not being part of another *Node* have a unique name in this standard, the symbolic name is unique. For example, the *ServerType* defined in 6.3.1 has the symbolic name "ServerType". One of its *InstanceDeclarations* would be identified as "ServerType.ServerCapabilities". Since this *Object* is complex, another *InstanceDeclaration* of the *ServerType* is "ServerType.ServerCapabilities.MinSupportedSampleRate". The *Server Object* defined in 8.3.2 is based on the *ServerType* and has the symbolic name "Server". Therefore, the instance based on the *InstanceDeclaration* described above has the symbolic name "Server.ServerCapabilities.MinSupportedSampleRate".

The *NamespaceIndex* for all *NodeIds* defined in this standard is 0. The namespace for this *NamespaceIndex* is specified in IEC 62541-3.

Note that this standard not only defines concrete *Nodes*, but also requires that some *Nodes* have to be generated, for example one for each Session running on the *Server*. The *NodeIds* of those *Nodes* are server-specific, including the Namespace. However the *NamespaceIndex* of those *Nodes* cannot be the *NamespaceIndex* 0, because they are not defined by the OPC Foundation but generated by the *Server*.

4.2 BrowseNames

The text part of the *BrowseNames* for all *Nodes* defined in this standard is specified in the tables defining the *Nodes*. The *NamespaceIndex* for all *BrowseNames* defined in this document is 0.

5 Common Attributes

5.1 General

For all *Nodes* specified in *NamespaceIndex* 0, the *Attributes* named in Table 3 shall be set as specified in Table 3.

Table 3 – Common Node Attributes

Attribute	Value
DisplayName	The <i>DisplayName</i> is a <i>LocalizedText</i> . Each server shall provide the <i>DisplayName</i> identical to the <i>BrowseName</i> of the <i>Node</i> for the LocaleId "en". Whether the server provides translated names for other LocaleIds is server-specific.
Description	Optionally a server-specific description is provided.
NodeClass	Shall reflect the <i>NodeClass</i> of the <i>Node</i> .
NodeId	The <i>NodeId</i> is described by <i>BrowseNames</i> as defined in 4.1 and defined in IEC 62541-6.
WriteMask	Optionally the <i>WriteMask Attribute</i> can be provided. If the <i>WriteMask Attribute</i> is provided, it shall set all non-server-specific <i>Attributes</i> to not writable. For example, the <i>Description Attribute</i> may be set to writable since a <i>Server</i> may provide a server-specific description for the <i>Node</i> . The <i>NodeId</i> shall not be writable, because it is defined for each <i>Node</i> in this standard.
UserWriteMask	Optionally the <i>UserWriteMask Attribute</i> can be provided. The same rules as for the <i>WriteMask Attribute</i> apply.
RolePermissions	Optionally server-specific role permissions can be provided.
UserRolePermissions	Optionally the role permissions of the current <i>Session</i> can be provided. The value is server-specific and depend on the <i>RolePermissions Attribute</i> (if provided) and the current <i>Session</i> .
AccessRestrictions	Optionally server-specific access restrictions can be provided.

5.2 Objects

For all *Objects* specified in *NamespaceIndex* 0, the *Attributes* named in Table 4 shall be set as specified in Table 4.

Table 4 – Common Object Attributes

Attribute	Value
EventNotifier	Whether the <i>Node</i> can be used to subscribe to <i>Events</i> or not is server-specific.

5.3 Variables

For all *Variables* specified in *NamespaceIndex* 0, the *Attributes* named in Table 5 shall be set as specified in Table 5.

Table 5 – Common Variable Attributes

Attribute	Value
MinimumSamplingInterval	Optionally, a server-specific minimum sampling interval is provided.
AccessLevel	The access level for <i>Variables</i> used for type definitions is server-specific, for all other <i>Variables</i> defined in this standard, the access level shall allow reading; other settings are server-specific.
UserAccessLevel	The value for the <i>UserAccessLevel Attribute</i> is server-specific. It is assumed that all <i>Variables</i> can be accessed by at least one user.
Value	For <i>Variables</i> used as <i>InstanceDeclarations</i> , the value is server-specific; otherwise it shall represent the value described in the text.
ArrayDimensions	If the <i>ValueRank</i> does not identify an array of a specific dimension (i.e. <i>ValueRank</i> ≤ 0) the <i>ArrayDimensions</i> can either be set to null or the <i>Attribute</i> is missing. This behaviour is server-specific. If the <i>ValueRank</i> specifies an array of a specific dimension (i.e. <i>ValueRank</i> > 0) then the <i>ArrayDimensions Attribute</i> shall be specified.
Historizing	The value for the <i>Historizing Attribute</i> is server-specific.
AccessLevelEx	If the <i>AccessLevelEx Attribute</i> is provided, it shall have the bits 8, 9, and 10 set to 0, meaning that read and write operations on an individual <i>Variable</i> are atomic, and arrays can be partly written.

5.4 VariableTypes

For all *VariableTypes* specified in *NamespaceIndex* 0, the *Attributes* named in Table 6 shall be set as specified in Table 6.

Table 6 – Common VariableType Attributes

Attributes	Value
Value	Optionally a server-specific default value can be provided.
ArrayDimensions	If the <i>ValueRank</i> does not identify an array of a specific dimension (i.e. <i>ValueRank</i> ≤ 0) the <i>ArrayDimensions</i> can either be set to null or the <i>Attribute</i> is missing. This behaviour is server-specific. If the <i>ValueRank</i> specifies an array of a specific dimension (i.e. <i>ValueRank</i> > 0) then the <i>ArrayDimensions Attribute</i> shall be specified.

5.5 Methods

For all *Methods* specified in *NamespaceIndex* 0, the *Attributes* named in Table 7 shall be set as specified in Table 7.

Table 7 – Common Method Attributes

Attributes	Value
Executable	All <i>Methods</i> defined in this specification shall be executable (<i>Executable Attribute</i> set to "True"), unless it is defined differently in the <i>Method</i> definition.
UserExecutable	The value of the <i>UserExecutable Attribute</i> is server-specific. It is assumed that all <i>Methods</i> can be executed by at least one user.

6 Standard ObjectTypes

6.1 General

Typically, the components of an *ObjectType* are fixed and can be extended by subtyping. However, since each *Object* of an *ObjectType* can be extended with additional components, this standard allows extending the standard *ObjectTypes* defined in this document with additional components. Thereby, it is possible to express the additional information in the type definition that would already be contained in each *Object*. Some *ObjectTypes* already provide entry points for server-specific extensions. However, it is not allowed to restrict the components of the standard *ObjectTypes* defined in this standard. An example of extending the *ObjectTypes* is putting the standard *Property NodeVersion* defined in IEC 62541-3 into the *BaseObjectType*, stating that each *Object* of the *Server* will provide a *NodeVersion*.

6.2 BaseObjectType

The *BaseObjectType* is used as type definition whenever there is an *Object* having no more concrete type definitions available. *Servers* should avoid using this *ObjectType* and use a more specific type, if possible. This *ObjectType* is the base *ObjectType* and all other *ObjectTypes* shall either directly or indirectly inherit from it. However, it might not be possible for *Servers* to provide all *HasSubtype References* from this *ObjectType* to its subtypes, and therefore it is not required to provide this information.

There are no *References* specified for this *ObjectType*. It is formally defined in Table 8.

Table 8 – BaseObjectType definition

Attribute	Value				
BrowseName	BaseObjectType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Conformance Units					
Base Info Base Types					

6.3 ObjectTypes for the Server Object

6.3.1 ServerType

This *ObjectType* defines the capabilities supported by the OPC UA *Server*. It is formally defined in Table 9.

205

Table 9 – ServerType definition

Attribute	Value			
BrowseName	ServerType			
IsAbstract	False			
References	NodeClass	BrowseName	DataType / TypeDefinition	Modelling Rule
Subtype of the BaseObjectType defined in 6.2				
HasProperty	Variable	ServerArray	String[] PropertyType	Mandatory
HasProperty	Variable	NamespaceArray	String[] PropertyType	Mandatory
HasProperty	Variable	UrisVersion	VersionTime PropertyType	Optional
HasComponent	Variable	ServerStatus ¹	ServerStatusDataType ServerStatusType	Mandatory
HasProperty	Variable	ServiceLevel	Byte PropertyType	Mandatory
HasProperty	Variable	Auditing	Boolean PropertyType	Mandatory
HasProperty	Variable	EstimatedReturnTime	DateTime PropertyType	Optional
HasProperty	Variable	LocalTime	TimeZoneDataType PropertyType	Optional
HasComponent	Object	ServerCapabilities ¹	- ServerCapabilitiesType	Mandatory
HasComponent	Object	ServerDiagnostics ¹	- ServerDiagnosticsType	Mandatory
HasComponent	Object	VendorServerInfo	- VendorServerInfoType	Mandatory
HasComponent	Object	ServerRedundancy ¹	- ServerRedundancyType	Mandatory
HasComponent	Object	Namespaces	- NamespacesType	Optional
HasComponent	Method	GetMonitoredItems	Defined in 9.1	Optional
HasComponent	Method	ResendData	Defined in 9.2	Optional
HasComponent	Method	SetSubscriptionDurable	Defined in 9.3	Optional
HasComponent	Method	RequestServerStateChange	Defined in 9.4	Optional
Conformance Units				
Base Info ServerType				

206

207 *ServerArray* defines an array of *Server* URIs. This *Variable* is also referred to as the *server*
 208 *table*. Each URI in this array represents a globally-unique logical name for a *Server* within the
 209 scope of the network in which it is installed. Each OPC UA *Server* instance has a single URI
 210 that is used in the *server table* of other OPC UA *Servers*. Index 0 is reserved for the URI of the
 211 local *Server*. Values above 0 are used to identify remote *Servers* and are specific to a *Server*.
 212 IEC 62541-4 describes discovery mechanism that can be used to resolve URIs into URLs. The
 213 *Server* URI is case sensitive.

214 The URI of the *ServerArray* with Index 0 shall be identical to the URI of the *NamespaceArray*
 215 with Index 1, since both represent the local *Server*.

216 The indexes into the *server table* are referred to as *server indexes* or *server names*. They are
 217 used in OPC UA *Services* to identify *TargetNodes* of *References* that reside in remote *Servers*.
 218 Clients may read the entire table or they may read individual entries in the table. The *Server*
 219 shall not modify or delete entries of this table while any client has an open session to the *Server*,
 220 because clients may cache the *server table*. A *Server* may add entries to the *server table* even
 221 if clients are connected to the *Server*.

222 *NamespaceArray* defines an array of namespace URIs. This *Variable* is also referred as
 223 *namespace table*. The indexes into the *namespace table* are referred to as *NamespaceIndexes*.
 224 *NamespaceIndexes* are used in *NodeIds* in OPC UA *Services*, rather than the longer
 225 namespace URI. Index 0 is reserved for the OPC UA namespace, and index 1 is reserved for

the local *Server*. Clients may read the entire *namespace table* or they may read individual entries in the *namespace table*. The *Server* shall not modify or delete entries of the *namespace table* while any client has an open session to the *Server*, because clients may cache the *namespace table*. A *Server* may add entries to the *namespace table* even if clients are connected to the *Server*. It is recommended that *Servers* not change the indexes of the *namespace table* but only add entries, because the client may cache *NodeIds* using the indexes. Nevertheless, it might not always be possible for *Servers* to avoid changing indexes in the *namespace table*. Clients that cache *NamespaceIndexes* of *NodeIds* should always check when starting a session to verify that the cached *NamespaceIndexes* have not changed.

UrisVersion defines the version of the *ServerArray* and the *NamespaceArray*. Everytime the *ServerArray* or the *NamespaceArray* is changed, the value of the *UrisVersion* shall be updated to a value greater than the previous value. The *UrisVersion Property* is used in combination with the *SessionlessInvoke Service* defined in IEC 62541-4. If a *Server* supports this *Service*, the *Server* shall support this *Property*. It is the responsibility of the *Server* to provide a consistent set of values for the *ServerArray*, *NamespaceArray* and the *UrisVersion Properties*. The *VersionTime DataType* is defined in IEC 62541-4.

ServerStatus contains elements that describe the status of the *Server*. See 12.10 for a description of its elements.

ServiceLevel describes the ability of the *Server* to provide its data to the *Client*. The value range is from 0 to 255, where 0 indicates the worst and 255 indicates the best. *Servers* shall set the *ServiceLevel* according to the sub-ranges defined in IEC 62541-4.

Auditing is a Boolean specifying if the *Server* is currently generating audit events. It is set to TRUE if the *Server* generates audit events, otherwise to false. The *Profiles* defined in IEC 62541-7 specify what kind of audit events are generated by the *Server*.

EstimatedReturnTime indicates the time at which the *Server* is expected to have a *ServerStatus.State* of *RUNNING_0*. A *Client* that observes a shutdown or a *ServiceLevel* of 0 should either wait until after this time to attempt to reconnect to this *Server* or enter into slow retry logic. For example, most *Clients* will attempt to reconnect after a failure immediately and then progressively increase the delay between attempts until some maximum delay. This time can be used to trigger the *Client* to start its reconnect logic with some delay.

LocalTime is a structure containing the Offset and the DaylightSavingInOffset flag. The Offset specifies the time difference (in minutes) between the *Server* time in UTC and the local time at the *Server* location. If DaylightSavingInOffset is TRUE, then Standard/Daylight savings time (DST) at the *Server* location is in effect and Offset includes the DST correction. If FALSE then the Offset does not include DST correction and DST may or may not be in effect.

ServerCapabilities defines the capabilities supported by the OPC UA *Server*. See 6.3.2 for its description.

ServerDiagnostics defines diagnostic information about the OPC UA *Server*. See 6.3.3 for its description.

VendorServerInfo represents the browse entry point for vendor-defined *Server* information. This *Object* is required to be present even if there are no vendor-defined *Objects* beneath it. See 6.3.6 for its description.

ServerRedundancy describes the redundancy capabilities provided by the *Server*. This *Object* is required even if the *Server* does not provide any redundancy support. If the *Server* supports redundancy, then a subtype of *ServerRedundancyType* is used to describe its capabilities. Otherwise, it provides an *Object* of type *ServerRedundancyType* with the *Property* *RedundancySupport* set to none. See 6.3.7 for the description of *ServerRedundancyType*.

Namespace provides a list of *NamespaceMetadataType Objects* with additional information about the namespaces used in the *Server*. See 6.3.14 for the description of *NamespaceMetadataType*.

276 The *GetMonitoredItems Method* is used to identify the *MonitoredItems* of a *Subscription*. It is
277 defined in 9.1; the intended usage is defined in IEC 62541-4.

278 The *ResendData Method* is used to get the latest values of the data monitored items of a
279 *Subscription*. It is defined in 9.2; the intended usage is defined in IEC 62541-4.

280 The *SetSubscriptionDurable Method* is used to set a *Subscription* into a mode where
281 *MonitoredItem* data and event queues are stored and delivered even if an OPC UA *Client* was
282 disconnected for a longer time or the OPC UA *Server* was restarted. It is defined in 9.3; the
283 intended usage is defined in IEC 62541-4.

284 The *RequestServerStateChange Method* allows a *Client* to request a state change in the
285 *Server*. It is defined in 9.4; the intended usage is defined in IEC 62541-4.

286

287 **6.3.2 ServerCapabilitiesType**

288 This *ObjectType* defines the capabilities supported by the OPC UA *Server*. It is formally defined
289 in Table 10.

290

Table 10 – ServerCapabilitiesType definition

Attribute	Value			
BrowseName	ServerCapabilitiesType			
IsAbstract	False			
References	NodeClass	BrowseName	DataType / TypeDefinition	ModellingRule
Subtype of the BaseObjectType defined in 6.2				
HasProperty	Variable	ServerProfileArray	String[] PropertyType	Mandatory
HasProperty	Variable	LocaleIdArray	LocaleId[] PropertyType	Mandatory
HasProperty	Variable	MinSupportedSampleRate	Duration PropertyType	Mandatory
HasProperty	Variable	MaxBrowseContinuationPoints	UInt16 PropertyType	Mandatory
HasProperty	Variable	MaxQueryContinuationPoints	UInt16 PropertyType	Mandatory
HasProperty	Variable	MaxHistoryContinuationPoints	UInt16 PropertyType	Mandatory
HasProperty	Variable	SoftwareCertificates	SignedSoftwareCertificate[] PropertyType	Mandatory
HasProperty	Variable	MaxArrayLength	UInt32 PropertyType	Optional
HasProperty	Variable	MaxStringLength	UInt32 PropertyType	Optional
HasProperty	Variable	MaxByteStringLength	UInt32 PropertyType	Optional
HasComponent	Object	OperationLimits	-- OperationLimitsType	Optional
HasComponent	Object	ModellingRules	-- FolderType	Mandatory
HasComponent	Object	AggregateFunctions	-- FolderType	Mandatory
HasComponent	Object	RoleSet	RoleSetType	Optional
HasProperty	Variable	MaxSessions	UInt32 PropertyType	Optional
HasProperty	Variable	MaxSubscriptions	UInt32 PropertyType	Optional
HasProperty	Variable	MaxMonitoredItems	UInt32 PropertyType	Optional
HasProperty	Variable	MaxSubscriptionsPerSession	UInt32 PropertyType	Optional
HasProperty	Variable	MaxMonitoredItemsPerSubscription	UInt32 PropertyType	Optional
HasProperty	Variable	MaxSelectClauseParameters	UInt32 PropertyType	Optional
HasProperty	Variable	MaxWhereClauseParameters	UInt32 PropertyType	Optional
HasProperty	Variable	MaxMonitoredItemsQueueSize	UInt32 PropertyType	Optional
HasComponent	Variable	<VendorCapability>	BaseDataType ServerVendorCapabilityType	OptionalPlaceholder
HasProperty	Variable	ConformanceUnits	QualifiedName[] PropertyType	Optional
Conformance Units				
Base Info ServerType				

291

292 *ServerProfileArray* lists the *Profiles* that the *Server* supports. The *String* shall be the URI of the

293 *Profile*. See IEC 62541-7 for definitions of OPC UA *Server Profiles*. This list should be limited

294 to the *Profiles* the *Server* supports in its current configuration.

295 *LocaleIdArray* is an array of *LocaleIds* that are known to be supported by the *Server*. The *Server*

296 might not be aware of all *LocaleIds* that it supports because it may provide access to underlying

297 servers, systems or devices that do not report the *LocaleIds* that they support.

298 *MinSupportedSampleRate* defines the minimum supported sample rate, including 0, which is
299 supported by the *Server*.

300 *MaxBrowseContinuationPoints* is an integer specifying the maximum number of parallel
301 continuation points of the *Browse Service* that the *Server* can support per session. The value
302 specifies the maximum the *Server* can support under normal circumstances, so there is no
303 guarantee the *Server* can always support the maximum. The client should not open more
304 *Browse* calls with open continuation points than exposed in this *Variable*. The value 0 indicates
305 that the *Server* does not restrict the number of parallel continuation points the client should use.

306 *MaxQueryContinuationPoints* is an integer specifying the maximum number of parallel
307 continuation points of the *QueryFirst Services* that the *Server* can support per session. The
308 value specifies the maximum the *Server* can support under normal circumstances, so there is
309 no guarantee the *Server* can always support the maximum. The client should not open more
310 *QueryFirst* calls with open continuation points than exposed in this *Variable*. The value 0
311 indicates that the *Server* does not restrict the number of parallel continuation points the client
312 should use.

313 *MaxHistoryContinuationPoints* is an integer specifying the maximum number of parallel
314 continuation points of the *HistoryRead Services* that the *Server* can support per session. The
315 value specifies the maximum the *Server* can support under normal circumstances, so there is
316 no guarantee the *Server* can always support the maximum. The client should not open more
317 *HistoryRead* calls with open continuation points than exposed in this *Variable*. The value 0
318 indicates that the *Server* does not restrict the number of parallel continuation points the client
319 should use.

320 *SoftwareCertificates* is an array of *SignedSoftwareCertificates* containing all
321 *SoftwareCertificates* supported by the *Server*. A *SoftwareCertificate* identifies capabilities of the
322 *Server*. It contains the list of *Profiles* supported by the *Server*. *Profiles* are described in IEC
323 62541-7.

324 The *MaxArrayLength Property* indicates the maximum length of a one or multidimensional array
325 supported by *Variables*, *Method* arguments and *Event* fields of the *Server*. In a multidimensional
326 array it indicates the overall length. For example, a three-dimensional array of 2x3x10 has the
327 array length of 60. The *Server* might further restrict the length for individual *Variables*, *Method*
328 arguments or *Event* fields without notice to the client. *Servers* may use the *Property*
329 *MaxArrayLength* defined in IEC 62541-3 on individual *DataVariables* to specify the size on
330 individual values. The individual *Property* may have a larger or smaller value than
331 *MaxArrayLength*.

332 The *MaxStringLength Property* indicates the maximum number of bytes in *Strings* and
333 *LocalizedText* supported by *Variables*, *Method* arguments and *Event* fields of the *Server*.
334 *Servers* may override this setting by adding the *MaxStringLength Property* defined in IEC
335 62541-3 to an individual *DataVariable*. If a *Server* does not impose a maximum number of bytes
336 or is not able to determine the maximum number of bytes this *Property* shall not be provided.

337 The *MaxByteStringLength Property* indicates the maximum number of bytes in a *ByteString*
338 supported by *Variables*, *Method* arguments and *Event* fields of the *Server*. *Servers* may override
339 this setting by adding the *MaxByteStringLength Property* defined in IEC 62541-3 to an individual
340 *DataVariable*. If a *Server* does not impose a maximum number of bytes or is not able to
341 determine the maximum number of bytes this *Property* shall not be provided.

342 *OperationLimits* is an entry point to access information on operation limits of the *Server*, for
343 example the maximum length of an array in a read *Service* call.

344 *ModellingRules* is an entry point to browse to all *ModellingRules* supported by the *Server*. All
345 *ModellingRules* supported by the *Server* should be able to be browsed starting from this *Object*.

346 *AggregateFunctions* is an entry point to browse to all *AggregateFunctions* supported by the
347 *Server*. All *AggregateFunctions* supported by the *Server* should be able to be browsed starting
348 from this *Object*. *AggregateFunctions* are *Objects* of *AggregateFunctionType*.

349 The *RoleSet Object* is used to publish all *Roles* supported by the *Server*. The *RoleSetType* is
350 specified in IEC 62541-18.

351 *MaxSessions* is an integer specifying the maximum number of concurrent *Sessions* the *Server*
352 can support. The value specifies the maximum the *Server* can support under normal
353 circumstances, therefore there is no guarantee the *Server* can always support the maximum.

354 *MaxSubscriptions* is an integer specifying the maximum number of *Subscriptions* the *Server*
355 can support. The value specifies the maximum the *Server* can support under normal
356 circumstances, therefore there is no guarantee the *Server* can always support the maximum.

357 *MaxMonitoredItems* is an integer specifying the maximum number of *MonitoredItems* the *Server*
358 can support. The value specifies the maximum the *Server* can support under normal
359 circumstances, therefore there is no guarantee the *Server* can always support the maximum.

360 *MaxSubscriptionsPerSession* is an integer specifying the maximum number of *Subscriptions*
361 per *Session* the *Server* can support. The value specifies the maximum the *Server* can support
362 under normal circumstances, therefore there is no guarantee the *Server* can always support the
363 maximum.

364 *MaxMonitoredItemsPerSubscription* is an integer specifying the maximum number of
365 *MonitoredItems* per *Subscription* the *Server* can support. The value specifies the maximum the
366 *Server* can support under normal circumstances, therefore there is no guarantee the *Server* can
367 always support the maximum.

368 *MaxSelectClauseParameters* is an integer specifying the maximum number of *EventField*
369 *SelectClause Parameters* the *Server* can support for an *EventFilter*. The value specifies the
370 maximum the *Server* can support under normal circumstances, therefore there is no guarantee
371 the *Server* can always support the maximum.

372 *MaxWhereClauseParameters* is an integer specifying the maximum number of *EventField*
373 *WhereClause Parameters* the *Server* can support for an *EventFilter*. The value specifies the
374 maximum the *Server* can support under normal circumstances, therefore there is no guarantee
375 the *Server* can always support the maximum.

376 *MaxMonitoredItemsQueueSize* is an integer specifying the maximum size of data *MonitoredItem*
377 queues. The value specifies the maximum the *Server* can support under normal circumstances,
378 therefore there is no guarantee the *Server* can always support the maximum.

379 When a vendor can not determine a limit of an optional component then the component shall
380 not be included.

381 When vendors expose their own capabilities, they should add additional *Nodes* to the standard
382 *ServerCapabilities Object* instance.

383 The *<VendorCapability> Variable* is used to provide information about the server-specific
384 capabilities.

385 *ConformanceUnits* is a *QualifiedName* array specifying the set of conformance units the *Server*
386 supports. This list should be limited to the *ConformanceUnits* the *Server* supports in its current
387 configuration.

388 6.3.3 ServerDiagnosticsType

389 This *ObjectType* defines diagnostic information about the OPC UA *Server*. This *ObjectType* is
390 formally defined in Table 11.

Table 11 – ServerDiagnosticsType definition

Attribute	Value			
BrowseName	ServerDiagnosticsType			
IsAbstract	False			
References	Node Class	BrowseName	DataType / TypeDefinition	Modelling Rule
Subtype of the BaseObjectType defined in 6.2				
HasComponent	Variable	ServerDiagnosticsSummary	ServerDiagnosticsSummaryDataType ServerDiagnosticsSummaryType	Mandatory
HasComponent	Variable	SamplingIntervalDiagnosticsArray	SamplingIntervalDiagnosticsDataType[] SamplingIntervalDiagnosticsArrayType	Optional
HasComponent	Variable	SubscriptionDiagnosticsArray	SubscriptionDiagnosticsDataType[] SubscriptionDiagnosticsArrayType	Mandatory
HasComponent	Object	SessionsDiagnosticsSummary	-- SessionsDiagnosticsSummaryType	Mandatory
HasProperty	Variable	EnabledFlag	Boolean PropertyType	Mandatory
Conformance Units				
Base Info ServerType				

ServerDiagnosticsSummary contains diagnostic summary information for the *Server*, as defined in 12.9.

SamplingIntervalDiagnosticsArray is an array of diagnostic information per sampling rate as defined in 12.8. There is one entry for each sampling rate currently used by the *Server*. Its *TypeDefinitionNode* is the *VariableType SamplingIntervalDiagnosticsArrayType*, providing a *Variable* for each entry in the array, as defined in 7.9.

The sampling interval diagnostics are only collected by *Servers* which use a fixed set of sampling intervals. In these cases, length of the array and the set of contained *Variables* will be determined by the *Server* configuration and the *NodeId* assigned to a given sampling interval diagnostics variable shall not change as long as the *Server* configuration does not change. A *Server* may not expose the *SamplingIntervalDiagnosticsArray* if it does not use fixed sampling rates.

SubscriptionDiagnosticsArray is an array of Subscription diagnostic information per subscription, as defined in 12.15. There is one entry for each Notification channel actually established in the *Server*. Its *TypeDefinitionNode* is the *VariableType SubscriptionDiagnosticsArrayType*, providing a *Variable* for each entry in the array as defined in 7.11. Those *Variables* are also used as *Variables* referenced by other *Variables*.

SessionsDiagnosticsSummary contains diagnostic information per session, as defined in 6.3.4.

EnabledFlag identifies whether or not diagnostic information is collected by the *Server*. It can also be used by a client to enable or disable the collection of diagnostic information of the *Server*. The following settings of the Boolean value apply: TRUE indicates that the *Server* collects diagnostic information, and setting the value to TRUE leads to resetting and enabling the collection. FALSE indicates that no diagnostic information is collected, and setting the value to FALSE disables the collection without resetting the diagnostic values.

When diagnostics are turned off, the *Server* can return *Bad_NodeIdUnknown* for all static diagnostic *Nodes* except the *EnabledFlag Property*. Dynamic diagnostic *Nodes* (such as the *Session Nodes*) will not appear in the *AddressSpace*.

If the collection of diagnostic information is not supported, the *EnabledFlag Property* will be read only.

6.3.4 SessionsDiagnosticsSummaryType

This *ObjectType* defines diagnostic information about the sessions of the OPC UA *Server*. This *ObjectType* is formally defined in Table 12.

Table 12 – SessionsDiagnosticsSummaryType definition

Attribute		Value		
BrowseName		SessionsDiagnosticsSummaryType		
IsAbstract		False		
References	NodeClass	BrowseName	Data Type / TypeDefinition	Modelling Rule
Subtype of the BaseObjectType defined in 6.2				
HasComponent	Variable	SessionDiagnosticsArray	SessionDiagnosticsDataType[] SessionDiagnosticsArrayType	Mandatory
HasComponent	Variable	SessionSecurityDiagnosticsArray	SessionSecurityDiagnosticsDataType[] SessionSecurityDiagnosticsArrayType	Mandatory
HasComponent	Object	<ClientName>	-- SessionDiagnosticsObjectType	Optional Placeholder
Conformance Units				
Base Info ServerType				

SessionDiagnosticsArray provides an array with an entry for each session in the *Server* having general diagnostic information about a session.

SessionSecurityDiagnosticsArray provides an array with an entry for each active session in the *Server* having security-related diagnostic information about a session. Since this information is security-related, it should not be made accessible to all users, but only to authorised users over encrypted channels.

For each session of the *Server*, this *Object* also provides an *Object* representing the session, indicated by <ClientName>. The BrowseName could be derived from the *sessionName* defined in the *CreateSession Service* (IEC 62541-4) or some other server-specific mechanisms. It is of the *ObjectType* SessionDiagnosticsObjectType, as defined in 6.3.5.

6.3.5 SessionDiagnosticsObjectType

This *ObjectType* defines diagnostic information about a session of the OPC UA *Server*. This *ObjectType* is formally defined in Table 13.

Table 13 – SessionDiagnosticsObjectType definition

Attribute		Value		
BrowseName		SessionDiagnosticsObjectType		
IsAbstract		False		
References	NodeClass	BrowseName	Data Type / TypeDefinition	Modelling Rule
Subtype of the BaseObjectType defined in 6.2				
HasComponent	Variable	SessionDiagnostics	SessionDiagnosticsDataType SessionDiagnosticsVariableType	Mandatory
HasComponent	Variable	SessionSecurityDiagnostics	SessionSecurityDiagnosticsDataType SessionSecurityDiagnosticsType	Mandatory
HasComponent	Variable	SubscriptionDiagnosticsArray	SubscriptionDiagnosticsDataType[] SubscriptionDiagnosticsArrayType	Mandatory
Conformance Units				
Base Info ServerType				

SessionDiagnostics contains general diagnostic information about the session; the *SessionSecurityDiagnostics Variable* contains security-related diagnostic information. Because the information of the second *Variable* is security-related, it should not be made accessible to all users, but only to authorised users.

SubscriptionDiagnosticsArray is an array of Subscription diagnostic information per opened subscription, as defined in 12.15. Its *TypeDefinitionNode* is the *VariableType* SubscriptionDiagnosticsArrayType providing a *Variable* for each entry in the array, as defined in 7.11.

6.3.6 VendorServerInfoType

This *ObjectType* defines a placeholder *Object* for vendor-specific information about the OPC UA Server. This *ObjectType* defines an empty *ObjectType* that has no components. It should be subtyped by vendors to define their vendor-specific information. This *ObjectType* is formally defined in Table 14.

Table 14 – VendorServerInfoType definition

Attribute	Value				
BrowseName	VendorServerInfoType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the BaseObjectType defined in 6.2					
Conformance Units					
Base Info ServerType					

6.3.7 ServerRedundancyType

This *ObjectType* defines the redundancy capabilities supported by the OPC UA Server. It is formally defined in Table 15.

Table 15 – ServerRedundancyType definition

Attribute	Value				
BrowseName	ServerRedundancyType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	Type Definition	Modelling Rule
Subtype of the BaseObjectType defined in 6.2					
HasProperty	Variable	RedundancySupport	RedundancySupport	PropertyType	Mandatory
HasProperty	Variable	RedundantServerArray	RedundantServerDataType[]	PropertyType	Optional
Conformance Units					
Base Info ServerType					

RedundancySupport indicates what redundancy is supported by the Server. Its values are defined in 12.5. It shall be set to NONE for all instances of the *ServerRedundancyType* using the *ObjectType* directly (no subtype).

The *RedundantServerArray* contains an array of available Servers in the *Redundant Set*; including their service levels (see 12.7). This array may change during a *Session*; the order of the array elements shall always be the same order as the array elements of the *ServerUriArray Variable*.

6.3.8 TransparentRedundancyType

This *ObjectType* is a subtype of *ServerRedundancyType* and is used to identify the capabilities of the OPC UA Server for server-controlled redundancy with a transparent switchover for the client. It is formally defined in Table 16.

Table 16 – TransparentRedundancyType definition

Attribute	Value				
BrowseName	TransparentRedundancyType				
IsAbstract	False				
References	Node Class	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the ServerRedundancyType defined in 6.3.7, i.e. inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	CurrentServerId	String	PropertyType	Mandatory
HasProperty	Variable	RedundantServerArray	RedundantServerDataType[]	PropertyType	Mandatory
Conformance Units					
Redundancy Server Transparent					

RedundancySupport is inherited from the *ServerRedundancyType*. It shall be set to *TRANSPARENT_4* for all instances of the *TransparentRedundancyType*.

Although, in a transparent switchover scenario, all redundant *Servers* serve under the same URI to the *Client*, it may be required to track the exact data source on the *Client*. Therefore, *CurrentServerId* contains an identifier of the currently-used *Server* in the *Redundant Set*. This *Server* is valid only inside a *Session*; if a *Client* opens several *Sessions*, different *Servers* of the redundant set of *Servers* may serve it in different *Sessions*. The value of the *CurrentServerId* may change due to *Failover* or load balancing, so a *Client* that needs to track its data source shall subscribe to this *Variable*.

As diagnostic information, the *RedundantServerArray* contains an array of available *Servers* in the *Redundant Set*; including their service levels (see 12.7). This array may change during a *Session*.

6.3.9 NonTransparentRedundancyType

This *ObjectType* is a subtype of *ServerRedundancyType* and is used to identify the capabilities of the OPC UA *Server* for non-transparent redundancy. It is formally defined in Table 17.

Table 17 – NonTransparentRedundancyType definition

Attribute	Value				
BrowseName	NonTransparentRedundancyType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the <i>ServerRedundancyType</i> defined in 6.3.7, which means it inherits the <i>InstanceDeclarations</i> of that Node.					
HasProperty	Variable	ServerUriArray	String[]	PropertyType	Mandatory
Conformance Units					
Redundancy Server					

RedundancySupport is inherited from the *ServerRedundancyType*. It shall be set to *COLD*, *WARM*, *HOT* or *HOT_AND_MIRRORED* for all instances of the *NonTransparentRedundancyType*. It defines the redundancy support provided by the *Server*. Its intended use is defined in IEC 62541-4.

6.3.10 NonTransparentNetworkRedundancyType

This *ObjectType* is a subtype of *NonTransparentRedundancyType* and is used to identify the capabilities of the OPC UA *Server* for non-transparent network redundancy. It is formally defined in Table 18.

Table 18 – NonTransparentNetworkRedundancyType definition

Attribute	Value				
BrowseName	NonTransparentNetworkRedundancyType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>NonTransparentRedundancyType</i> defined in 6.3.9, which means it inherits the <i>InstanceDeclarations</i> of that Node.					
HasProperty	Variable	ServerNetworkGroups	NetworkGroupDataType[]	PropertyType	Mandatory
Conformance Units					
Redundancy Server					

Clients switching between network paths to the same *Server* behave the same as *HotAndMirrored* redundancy. *Server* and network redundancy can be combined. In the combined approach it is important for the *Client* to know which *ServerUris* belong to the same *Server* representing different network paths and which *ServerUris* represent different *Servers*. Therefore, a *Server* implementing non-transparent network redundancy shall use the *NonTransparentNetworkRedundancyType* to identify its redundancy support.

RedundancySupport is inherited from the *ServerRedundancyType*. It shall be set to COLD, WARM, HOT or HOT_AND_MIRRORED for all instances of the *NonTransparentNetworkRedundancyType*. If no *Server* redundancy is supported (the *ServerUriArray* only contains one entry), the *RedundancySupport* shall be set to HOT_AND_MIRRORED.

The *ServerNetworkGroups* contains an array of *NetworkGroupDataType*. The URIs of the *Servers* in that array (in the *serverUri* of the structure) shall be exactly the same as the ones provided in the *ServerUriArray*. However, the order might be different. Thus the array represents a list of HotAndMirrored redundant *Servers*. If a *Server* only supports network redundancy, it has only one entry in the *ServerNetworkGroups*. The *networkPaths* in the structure represents the redundant network paths for each of the *Servers*. The *networkPaths* describes the different paths (one entry for each path) ordered by priority. Each network path contains an *endpointUrlList* having an array of Strings each containing a URL of an *Endpoint*. This allows using different protocol options for the same network path.

The *Endpoints* provided shall match with the *Endpoints* provided by the *GetEndpoints Service* of the corresponding *Server*.

6.3.11 OperationLimitsType

This *ObjectType* is a subtype of *FolderType* and is used to identify the operation limits of the OPC UA *Server*. It is formally defined in Table 19.

Table 19 – OperationLimitsType definition

Attribute	Value				
BrowseName	OperationLimitsType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>FolderType</i> defined in 6.6, which means it inherits the InstanceDeclarations of that Node.					
HasProperty	Variable	MaxNodesPerRead	UInt32	PropertyType	Optional
HasProperty	Variable	MaxNodesPerHistoryReadData	UInt32	PropertyType	Optional
HasProperty	Variable	MaxNodesPerHistoryReadEvents	UInt32	PropertyType	Optional
HasProperty	Variable	MaxNodesPerWrite	UInt32	PropertyType	Optional
HasProperty	Variable	MaxNodesPerHistoryUpdateData	UInt32	PropertyType	Optional
HasProperty	Variable	MaxNodesPerHistoryUpdateEvents	UInt32	PropertyType	Optional
HasProperty	Variable	MaxNodesPerMethodCall	UInt32	PropertyType	Optional
HasProperty	Variable	MaxNodesPerBrowse	UInt32	PropertyType	Optional
HasProperty	Variable	MaxNodesPerRegisterNodes	UInt32	PropertyType	Optional
HasProperty	Variable	MaxNodesPerTranslateBrowsePathsToNodeIds	UInt32	PropertyType	Optional
HasProperty	Variable	MaxNodesPerNodeManagement	UInt32	PropertyType	Optional
HasProperty	Variable	MaxMonitoredItemsPerCall	UInt32	PropertyType	Optional
Conformance Units					
Base Info ServerType					

Any operational limits *Property* that is provided shall have a non zero value.

The *MaxNodesPerRead Property* indicates the maximum size of the nodesToRead array when a *Client* calls the Read Service.

The *MaxNodesPerHistoryReadData Property* indicates the maximum size of the nodesToRead array when a *Client* calls the HistoryRead Service using the historyReadDetails RAW, PROCESSED, MODIFIED, or ATTIME.

The *MaxNodesPerHistoryReadEvents Property* indicates the maximum size of the nodesToRead array when a *Client* calls the HistoryRead Service using the historyReadDetails EVENTS.

The *MaxNodesPerWrite Property* indicates the maximum size of the nodesToWrite array when a *Client* calls the Write Service.

The *MaxNodesPerHistoryUpdateData Property* indicates the maximum size of the historyUpdateDetails array supported by the *Server* when a *Client* calls the HistoryUpdate Service.

The *MaxNodesPerHistoryUpdateEvents Property* indicates the maximum size of the historyUpdateDetails array when a *Client* calls the HistoryUpdate Service.

The *MaxNodesPerMethodCall Property* indicates the maximum size of the methodsToCall array when a *Client* calls the Call Service.

The *MaxNodesPerBrowse Property* indicates the maximum size of the nodesToBrowse array when calling the Browse Service or the continuationPoints array when a *Client* calls the BrowseNext Service.

The *MaxNodesPerRegisterNodes Property* indicates the maximum size of the nodesToRegister array when a *Client* calls the RegisterNodes Service and the maximum size of the nodesToUnregister when calling the UnregisterNodes Service.

The *MaxNodesPerTranslateBrowsePathsToNodeIds Property* indicates the maximum size of the browsePaths array when a *Client* calls the TranslateBrowsePathsToNodeIds Service.

The *MaxNodesPerNodeManagement Property* indicates the maximum size of the nodesToAdd array when a *Client* calls the AddNodes Service, the maximum size of the referencesToAdd array when a *Client* calls the AddReferences Service, the maximum size of the nodesToDelete array when a *Client* calls the DeleteNodes Service, and the maximum size of the referencesToDelete array when a *Client* calls the DeleteReferences Service.

The *MaxMonitoredItemsPerCall Property* indicates

- the maximum size of the itemsToCreate array when a *Client* calls the CreateMonitoredItems Service,
- the maximum size of the itemsToModify array when a *Client* calls the ModifyMonitoredItems Service,
- the maximum size of the monitoredItemIds array when a *Client* calls the SetMonitoringMode Service or the DeleteMonitoredItems Service,
- the maximum size of the sum of the linksToAdd and linksToRemove arrays when a *Client* calls the SetTriggering Service.

6.3.12 AddressSpaceFileType

This *ObjectType* defines the file for a namespace provided by the OPC UA Server. It is formally defined in Table 20. It represents an XML address space file using the XML schema defined in IEC 62541-6.

Table 20 – AddressSpaceFileType definition

Attribute	Value				
BrowseName	AddressSpaceFileType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the FileType defined in IEC 62541-20					
HasComponent	Method	ExportNamespace	The method has no parameters.		Optional
Conformance Units					
Base Info ServerType					

The *ExportNamespace Method* provides a way to export the namespace from the *Server AddressSpace* to the XML file represented by the *AddressSpaceFileType*. *Value Attributes* are only exported if they represent static configuration information. The client is expected to call

the *ExportNamespace Method* first to update the XML file and then access the file with the *Methods* defined in the *FileType*.

Servers might provide some vendor-specific mechanisms importing parts of an address space as subtype of this *ObjectType*, for example by defining appropriate *Methods*.

6.3.13 NamespaceMetadataType

This *ObjectType* defines the metadata for a namespace provided by the *Server*. It is formally defined in Table 21.

Instances of this *Object* allow *Servers* to provide more information like version information in addition to the namespace URI. Important information for aggregating *Servers* is provided by the *StaticNodeIdTypes*, *StaticNumericNodeIdRange* and *StaticStringNodeIdPattern Properties*.

Table 21 – NamespaceMetadataType definition

Attribute	Value				
BrowseName	NamespaceMetadataType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the BaseObjectType defined in 6.2					
HasProperty	Variable	NamespaceUri	String	PropertyType	Mandatory
HasProperty	Variable	NamespaceVersion	String	PropertyType	Mandatory
HasProperty	Variable	NamespacePublicationDate	DateTime	PropertyType	Mandatory
HasProperty	Variable	IsNamespaceSubset	Boolean	PropertyType	Mandatory
HasProperty	Variable	StaticNodeIdTypes	IdType[]	PropertyType	Mandatory
HasProperty	Variable	StaticNumericNodeIdRange	NumericRange[]	PropertyType	Mandatory
HasProperty	Variable	StaticStringNodeIdPattern	String	PropertyType	Mandatory
HasComponent	Object	NamespaceFile	-	AddressSpaceFileType	Optional
HasProperty	Variable	DefaultRolePermissions	RolePermissionType[]	PropertyType	Optional
HasProperty	Variable	DefaultUserRolePermissions	RolePermissionType[]	PropertyType	Optional
HasProperty	Variable	DefaultAccessRestrictions	AccessRestrictionType	PropertyType	Optional
HasProperty	Variable	ConfigurationVersion	VersionTime	PropertyType	Optional
HasProperty	Variable	ModelVersion	SemanticVersionString	PropertyType	Optional
Conformance Units					
Base Info ServerType					

The *BrowseName* of instances of this type shall be derived from the represented namespace. This can, for example, be done by using the index of the namespace in the *NamespaceArray* as *NamespaceIndex* of the *QualifiedName* and the namespace URI as *name* of the *QualifiedName*.

The *NamespaceUri Property* contains the namespace represented by an instance of the *MetaDataType*.

The *NamespaceVersion Property* provides version information for the namespace. It is intended for display purposes and shall not be used to programmatically identify the latest version. If there is no formal version defined for the namespace this *Property* shall be set to a null *String*.

The *NamespacePublicationDate Property* provides the publication date of the namespace version. This *Property* value can be used by *Clients* to determine the latest version if different versions are provided by different *Servers* when the *ModelVersion* is not provided. If there is no formal publication date defined for the namespace this *Property* shall be set to a null *DateTime*.

The *IsNamespaceSubset Property* defines whether all *Nodes* of the namespace are accessible in the *Server* or only a subset. It is set to FALSE if the full namespace is provided and TRUE if not. If the completeness is unknown then this *Property* shall be set to TRUE.

Static *Nodes* are identical for all *Attributes* in all *Servers*, including the *Value Attribute*. For *TypeDefinitionNodes*, also the *InstanceDeclarations* shall be identical. That means that for static *Nodes* the semantic is always the same. Namespaces with static *Nodes* are for example namespaces defined by standard bodies like the OPC Foundation. This is important information for aggregating *Servers*. If the namespace is dynamic and used in several *Servers* the aggregating *Server* needs to distinguish the namespace for each aggregated *Server*. The static *Nodes* of a namespace only need to be handled once, even if they are used by several aggregated *Servers*.

The *StaticNodeIdTypes Property* provides a list of *IdTypes* used for static *Nodes*. All *Nodes* in the *AddressSpace* of the namespace using one of the *IdTypes* in the array shall be static *Nodes*.

The *StaticNumericNodeIdRange Property* provides a list of *NumericRanges* used for numeric *NodeIds* of static *Nodes*. If the *StaticNodeIdTypes Property* contains an entry for numeric *NodeIds* then this *Property* is ignored.

The *StaticStringNodeIdPattern Property* provides a regular expression as defined for the *Like Operator* defined in IEC 62541-4 to filter for string *NodeIds* of static *Nodes*. If the *StaticNodeIdTypes Property* contains an entry for string *NodeIds* then this *Property* is ignored.

The *Object NamespaceFile* contains all *Nodes* and *References* of the namespace in an XML file where the Information Model XML Schema is defined in IEC 62541-6. The XML file is provided through an *AddressSpaceFileType Object*.

The *DefaultRolePermissions Property* provides the default permissions if a *Server* supports *RolePermissions* for the *Namespace*. A *Node* in the *Namespace* overrides this default by adding a *RolePermissions Attribute* to the *Node*. If a *Server* implements a vendor-specific *RolePermissions* model for a *Namespace*, it does not add the *DefaultRolePermissions Property* to the *NamespaceMetadata Object*.

The *DefaultUserRolePermissions Property* provides the default user permissions if a *Server* supports *UserRolePermissions* for the *Namespace*. A *Node* in the *Namespace* overrides this default by adding a *UserRolePermissions Attribute* to the *Node*. If a *Server* implements a vendor-specific *UserRolePermissions* model for a *Namespace*, it does not add the *DefaultUserRolePermissions Property* to the *NamespaceMetadata Object*.

The *DefaultAccessRestrictions Property* is present if a *Server* supports *AccessRestrictions* for the *Namespace* and provides the defaults. A *Node* in the *Namespace* overrides this default by adding a *AccessRestrictions Attribute* to the *Node*. If a *Server* implements a vendor-specific *AccessRestriction* model for a *Namespace*, it does not add the *DefaultAccessRestrictions Property* to the *NamespaceMetadata Object*.

The optional *ConfigurationVersion Property* indicates the current configuration of all *Nodes* of this *Namespace*. A *Node* of this *Namespace* is identified by having the corresponding *NamespaceIndex* in the *NodeId*. A *Client* can use the *ConfigurationVersion Property* in conjunction with the *Constant Field* of the *AccessLevelEx Attribute*, defined in IEC 62541-3, to optimize the use of *Variables* which have a constant value.

The optional *ModelVersion Property* indicates the *InformationModel* version expressed as a *SemanticVersionString* allowing programmatic comparisons.

6.3.14 NamespacesType

This *ObjectType* defines a list of *NamespaceMetadataType Objects* provided by the *Server*. It is formally defined in Table 22.

Table 22 – NamespacesType definition

Attribute	Value				
BrowseName	NamespacesType				
IsAbstract	False				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modelling Rule
Subtype of the BaseObjectType defined in 6.2					
HasComponent	Object	<NamespaceIdentifier>	-	NamespaceMetadataType	OptionalPlaceholder
Conformance Units					
Base Info ServerType					

The *ObjectType* contains a list of *NamespaceMetadataType Objects* representing the namespaces in the *Server*. The *BrowseName* of an *Object* shall be derived from the namespace represented by the *Object*. This can, for example, be done by using the index of the namespace in the *NamespaceArray* as *NamespaceIndex* of the *QualifiedName* and the namespace URI as *name* of the *QualifiedName*. *Clients* should not assume that all namespaces provided by a *Server* are present in this list as a namespace may not provide the information necessary to fill all mandatory *Properties* of the *NamespaceMetadataType*.

6.3.15 NonTransparentBackupRedundancyType

This *ObjectType* is a subtype of *NonTransparentRedundancyType* intended to be used to identify the capabilities of the *Servers* which support non-transparent primary with backup redundancy. It is formally defined in Table 23.

A *Server* represented by this *Type* shall use the *Warm Failover* mode defined in IEC 62541-4.

Table 23 – NonTransparentBackupRedundancyType definition

Attribute	Value				
BrowseName	NonTransparentBackupRedundancyType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the NonTransparentRedundancyType defined in 6.3.9.					
HasProperty	Variable	RedundantServerArray	RedundantServerDataType[]	PropertyType	Mandatory
HasProperty	Variable	Mode	RedundantServerMode	PropertyType	Mandatory
HasComponent	Method	Failover	The method has no parameters.		Mandatory
Conformance Units					
Redundancy Server					

The *Mode Property* represents the current operating mode of a *Server* in a non-transparent redundant server set. It is an enumeration defined in 12.44. This *Property* is used to indicate the availability status of backup *Servers*. A *Server* operating in primary mode provides the availability status of backup *Servers* and a *Server* operating in backup mode provides its ability to take over the primary operating mode. A *Server* operating in *BACKUP_READY*, *BACKUP_NOT_READY* or *PRIMARY_ONLY* mode shall use a *ServiceLevel* in the degraded range. A *Server* shall use a higher *ServiceLevel* value for *BACKUP_READY* than *BACKUP_NOT_READY*.

The *Failover Method* is used to request a manual transfer of primary and backup modes. This method has no arguments. This *Method* call may be directed to a primary or a backup *Server*. When called on the primary and more than one backup is ready to take over the primary will use some vendor specific logic to determine which backup will assume the primary mode. When called on a backup which is ready to take over, it will become the primary. The *Server* should set the *Executable Attribute* of this method to indicate if the failover is currently possible.

Signature

```
Failover (
);
```

Method Result Codes (defined in Call Service)

Result Code	Description
Bad_InvalidState	Defined in IEC 62541-4 This is returned when the failover failed due to the current <i>RedundantServerMode</i> mode of a primary being PRIMARY_ONLY or a backup being BACKUP_NOT_READY.
Bad_UserAccessDenied	Defined in IEC 62541-4

6.4 ObjectTypes used as EventTypes

6.4.1 General

This document defines standard *EventTypes*. They are represented in the *AddressSpace* as *ObjectTypes*. The *EventTypes* are already defined in IEC 62541-3. The following subclauses specify their representation in the *AddressSpace*.

6.4.2 BaseEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 24.

Table 24 – BaseEventType definition

Attribute	Value				
BrowseName	BaseEventType				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the <i>BaseObjectType</i> defined in 6.2					
HasProperty	Variable	EventId	ByteString	PropertyType	Mandatory
HasProperty	Variable	EventType	NodeId	PropertyType	Mandatory
HasProperty	Variable	SourceNode	NodeId	PropertyType	Mandatory
HasProperty	Variable	SourceName	String	PropertyType	Mandatory
HasProperty	Variable	Time	UtcTime	PropertyType	Mandatory
HasProperty	Variable	ReceiveTime	UtcTime	PropertyType	Mandatory
HasProperty	Variable	LocalTime	TimeZoneDataType	PropertyType	Optional
HasProperty	Variable	Message	LocalizedText	PropertyType	Mandatory
HasProperty	Variable	Severity	UInt16	PropertyType	Mandatory
HasProperty	Variable	ConditionClassId	NodeId	PropertyType	Optional
HasProperty	Variable	ConditionClassName	LocalizedText	PropertyType	Optional
HasProperty	Variable	ConditionSubClassId	NodeId[]	PropertyType	Optional
HasProperty	Variable	ConditionSubClassName	LocalizedText[]	PropertyType	Optional
Conformance Units					
Address Space Events					

EventId is generated by the *Server* to uniquely identify a particular *Event Notification*. The *Server* is responsible to ensure that each *Event* has its unique *EventId*. It may do this, for example, by putting GUIDs into the *ByteString*. Clients can use the *EventId* to assist in minimizing or eliminating gaps and overlaps that may occur during a redundancy failover. The *EventId* shall always be returned as value and the *Server* is not allowed to return a *StatusCode* for the *EventId* indicating an error.

EventType describes the specific type of *Event*. The *EventType* shall always be returned as value and the *Server* is not allowed to return a *StatusCode* for the *EventType* indicating an error.

The *SourceNode Property* identifies the *Node* that the *Event* originated from. If the *Event* is not specific to a *Node* the *NodeId* is set to null. Some subtypes of this *BaseEventType* may define additional rules for the *SourceNode Property*.

SourceName provides a description of the source of the *Event*. This could be the string-part of the *DisplayName* of the *Event* source using the default locale of the server, if the *Event* is specific to a *Node*, or some server-specific notation.

Time provides the time the *Event* occurred. This value is set as close to the event generator as possible. It often comes from the underlying system or device. Once set, intermediate OPC UA Servers shall not alter the value.

ReceiveTime provides the time the OPC UA Server received the *Event* from the underlying device of another Server. *ReceiveTime* is analogous to *ServerTimestamp* defined in IEC 62541-4, i.e. in the case where the OPC UA Server gets an *Event* from another OPC UA Server, each Server applies its own *ReceiveTime*. That implies that a *Client* may get the same *Event*, having the same *EventId*, from different Servers having different values of the *ReceiveTime*. The *ReceiveTime* shall always be returned as value and the Server is not allowed to return a *StatusCode* for the *ReceiveTime* indicating an error.

LocalTime is a structure containing the Offset and the DaylightSavingInOffset flag. The Offset specifies the time difference (in minutes) between the *Time Property* and the time at the location in which the event was issued. If DaylightSavingInOffset is TRUE, then Standard/Daylight savings time (DST) at the originating location is in effect and Offset includes the DST correction. If FALSE then the Offset does not include DST correction and DST may or may not have been in effect.

Message provides a human-readable and localizable text description of the *Event*. The Server may return any appropriate text to describe the *Event*. A null string is not a valid value; if the Server does not have a description, it shall return the string part of the *BrowseName* of the *Node* associated with the *Event*.

Severity is an indication of the urgency of the *Event*. This is also commonly called "priority". Values will range from 1 to 1 000, with 1 being the lowest severity and 1 000 being the highest. Typically, a severity of 1 would indicate an *Event* which is informational in nature, while a value of 1 000 would indicate an *Event* of catastrophic nature, which could potentially result in severe financial loss or loss of life.

It is expected that very few Server implementations will support 1 000 distinct severity levels. Therefore, Server developers are responsible for distributing their severity levels across the 1 to 1 000 range in such a manner that clients can assume a linear distribution. For example, a client wishing to present five severity levels to a user should be able to do the following mapping:

Client Severity	OPC Severity
HIGH	801 to 1 000
MEDIUM HIGH	601 to 800
MEDIUM	401 to 600
MEDIUM LOW	201 to 400
LOW	1 to 200

In many cases a strict linear mapping of underlying source severities to the OPC Severity range is not appropriate. The Server developer will instead intelligently map the underlying source severities to the 1 to 1 000 OPC Severity range in some other fashion. In particular, it is recommended that Server developers map *Events* of high urgency into the OPC severity range of 667 to 1 000, *Events* of medium urgency into the OPC severity range of 334 to 666 and *Events* of low urgency into OPC severities of 1 to 333.

For example, if a source supports 16 severity levels that are clustered such that severities 0 to 2 are considered to be LOW, 3 to 7 are MEDIUM and 8 to 15 are HIGH, then an appropriate mapping might be as follows:

OPC Range	Source Severity	OPC Severity
HIGH (667 to 1 000)	15	1 000
	14	955
	13	910
	12	865
	11	820
	10	775
	9	730
	8	685
MEDIUM (334 to 666)	7	650
	6	575
	5	500
	4	425
	3	350
LOW (1 to 333)	2	300
	1	150
	0	1

754

755 Some *Servers* might not support any *Events* which are catastrophic in nature, so they may
 756 choose to map all of their severities into a subset of the 1 to 1 000 range (for example, 1 to
 757 666). Other *Servers* might not support any *Events* which are merely informational, so they may
 758 choose to map all of their severities into a different subset of the 1 to 1 000 range (for example,
 759 334 to 1 000).

760 The purpose of this approach is to allow clients to use severity values from multiple *Servers*
 761 from different vendors in a consistent manner. Additional discussions of severity can be found
 762 in IEC 62541-9.

763 *ConditionClassId* specifies in which domain this *Event* is used. It is the *NodeId* of the
 764 corresponding subtype of *BaseConditionClassType*. See IEC 62541-9 for the definition of
 765 *ConditionClass* and a set of *ConditionClasses* defined in this standard. When using this
 766 *Property* for filtering, *Clients* have to specify all individual subtypes of *BaseConditionClassType*
 767 *NodeIds*. The *OfType* operator cannot be applied. *BaseConditionClassType* is used as class
 768 whenever an *Event* cannot be assigned to a more concrete class.

769 *ConditionClassName* provides the *DisplayName* of the of the *ConditionClassType* provided in
 770 the *ConditionClassId*.

771 *ConditionSubClassId* specifies additional class[es] that apply to the *Event*. It is the *NodeId* of
 772 the corresponding subtype of *BaseConditionClassType*. See IEC 62541-9 for the definition of
 773 *ConditionClass* and a set of *ConditionClasses* defined in this standard. When using this
 774 *Property* for filtering, *Clients* have to specify all individual sub types of *BaseConditionClassType*
 775 *NodeIds*. The *OfType* operator cannot be applied. The *Client* specifies a NULL in the filter, to
 776 return *Events* where no sub class is applied. When returning *Events*, if this optional field is not
 777 available in an *Event*, a NULL shall be returned for the field.

778 *ConditionSubClassName* provides the *DisplayName[s]* of the *ConditionClassType[s]* listed in
 779 the *ConditionSubClassId*.

780 The *BrowseNames* of the *ConditionClassId*, *ConditionClassName*, *ConditionSubClassId*, and
 781 *ConditionSubClassName* *Properties* originated from IEC 62541-9. They have now been
 782 generalized to also apply to *Events*, however to preserve backward compatibility the names
 783 have been preserved.

784

785 6.4.3 AuditEventType

786 This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally
 787 defined in Table 25.

Table 25 – AuditEventType definition

Attribute		Value			
BrowseName		AuditEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the <i>BaseEventType</i> defined in 6.4.2, which means it inherits the InstanceDeclarations of that Node.					
HasProperty	Variable	ActionTimeStamp	UtcTime	PropertyType	Mandatory
HasProperty	Variable	Status	Boolean	PropertyType	Mandatory
HasProperty	Variable	ServerId	String	PropertyType	Mandatory
HasProperty	Variable	ClientAuditEntryId	String	PropertyType	Mandatory
HasProperty	Variable	ClientUserId	String	PropertyType	Mandatory
Conformance Units					
Auditing Connections					
Auditing NodeManagement					
Auditing History Services					
Auditing Write					
Auditing Method					

This *EventType* inherits all *Properties* of the *BaseEventType*. Their semantic is defined in 6.4.2.

ActionTimeStamp identifies the time the user initiated the action that resulted in the *AuditEvent* being generated. It differs from the *Time Property* because this is the time the server generated the *AuditEvent* documenting the action.

Status identifies whether the requested action could be performed (set *Status* to TRUE) or not (set *Status* to FALSE).

ServerId uniquely identifies the *Server* generating the *Event*. It identifies the *Server* uniquely even in a server-controlled transparent redundancy scenario where several *Servers* may use the same URI.

ClientAuditEntryId contains the human-readable *AuditEntryId* defined in IEC 62541-4. If the *Server* is unable to decrypt *AuditEntryId* due to a certificate check failure, then some *Client* identification should be used such as the *Client*'s IP Address, port, MAC address and/or DNS name as the *ClientAuditEntryId*.

The *ClientUserId* identifies the user of the client requesting an action. The *ClientUserId* can be obtained from the *UserIdentityToken* passed in the *ActivateSession* call. If the *UserIdentityToken* is a *UserNameIdentityToken* then the *ClientUserId* is the *UserName*. If the *UserIdentityToken* is an *X509IdentityToken* then the *ClientUserId* is the X509 Subject Name of the *Certificate*. If the *UserIdentityToken* is an *IssuedIdentityToken* then the *ClientUserId* shall be a string that represents the owner of the token. The best choice for the string depends on the type of *IssuedIdentityToken*. If an *AnonymousIdentityToken* was used, the value is null. For a *SessionlessInvoke Service* invocation the *ClientUserId* shall be a string that represents the owner of the *authenticationToken* in the *requestHeader* of the *Service* invocation such as the name field of a JWT *IssuedIdentityToken*.

6.4.4 AuditSecurityEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 26.

Table 26 – AuditSecurityEventType definition

Attribute	Value				
BrowseName	AuditSecurityEventType				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the AuditEventType defined in 6.4.3, which means it inherits the InstanceDeclarations of that Node.					
HasProperty	Variable	StatusCodeId	StatusCode	PropertyType	Optional
Conformance Units					
Auditing Connections					

This *EventType* inherits all *Properties* of the *AuditEventType*. Their semantic is defined in 6.4.3.

The optional *StatusCodeId Property* provides the exact security error responsible for producing the *Event*.

6.4.5 AuditChannelEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 27.

Table 27 – AuditChannelEventType definition

Attribute	Value				
BrowseName	AuditChannelEventType				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the AuditSecurityEventType defined in 6.4.4, which means it inherits the InstanceDeclarations of that Node.					
HasProperty	Variable	SecureChannelId	String	PropertyType	Mandatory
Conformance Units					
Auditing Connections					

This *EventType* inherits all *Properties* of the *AuditSecurityEventType*. Their semantic is defined in 6.4.4. The *SourceNode Property* for *Events* of this type shall be assigned to the *Server Object*. The *SourceName* for *Events* of this type shall be "SecureChannel/" and the *Service* that generates the *Event* (e.g. *SecureChannel/OpenSecureChannel* or *SecureChannel/CloseSecureChannel*). If the *ClientUserId* is not available for a *CloseSecureChannel* call, then this parameter shall be set to "System/CloseSecureChannel".

The *SecureChannelId* shall uniquely identify the *SecureChannel*. The application shall use the same identifier in all *AuditEvents* related to the *Session Service Set* (*AuditCreateSessionEventType*, *AuditActivateSessionEventType* and their subtypes) and the *SecureChannel Service Set* (*AuditChannelEventType* and its subtypes).

6.4.6 AuditOpenSecureChannelEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 28.

Table 28 – AuditOpenSecureChannelEventType definition

Attribute		Value			
BrowseName		AuditOpenSecureChannelEventType			
IsAbstract		True			
References	Node Class	BrowseName	Data Type	Type Definition	Modelling Rule
Subtype of the <i>AuditChannelEventType</i> defined in 6.4.5, which means it inherits the InstanceDeclarations of that Node.					
HasProperty	Variable	ClientCertificate	ByteString	PropertyType	Mandatory
HasProperty	Variable	ClientCertificateThumbprint	String	PropertyType	Mandatory
HasProperty	Variable	RequestType	SecurityTokenRequestType	PropertyType	Mandatory
HasProperty	Variable	SecurityPolicyUri	String	PropertyType	Mandatory
HasProperty	Variable	SecurityMode	MessageSecurityMode	PropertyType	Mandatory
HasProperty	Variable	RequestedLifetime	Duration	PropertyType	Mandatory
HasProperty	Variable	CertificateErrorEventId	ByteString	PropertyType	Optional
Conformance Units					
Auditing Connections					

This *EventType* inherits all *Properties* of the *AuditChannelEventType*. Their semantic is defined in 6.4.5. The *SourceName* for *Events* of this type shall be "SecureChannel/OpenSecureChannel". The *ClientUserId* is not available for this call, thus this parameter shall be set to "System/OpenSecureChannel".

The additional *Properties* defined for this *EventType* reflect parameters of the *Service* call that triggers the *Event*.

ClientCertificate is the clientCertificate parameter of the OpenSecureChannel *Service* call.

ClientCertificateThumbprint is a thumbprint of the *ClientCertificate*. See IEC 62541-6 for details on thumbprints.

RequestType is the requestType parameter of the OpenSecureChannel *Service* call.

SecurityPolicyUri is the securityPolicyUri parameter of the OpenSecureChannel *Service* call.

SecurityMode is the securityMode parameter of the OpenSecureChannel *Service* call.

RequestedLifetime is the requestedLifetime parameter of the OpenSecureChannel *Service* call.

CertificateErrorEventId is the *EventId* of the corresponding *AuditCertificateEventType Event* used to report a certificate error.

6.4.7 AuditSessionEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 29.

Table 29 – AuditSessionEventType definition

Attribute		Value			
BrowseName		AuditSessionEventType			
IsAbstract		True			
References	Node Class	BrowseName	Data Type	Type Definition	Modelling Rule
Subtype of the <i>AuditSecurityEventType</i> defined in 6.4.4, which means it inherits the InstanceDeclarations of that Node.					
HasProperty	Variable	SessionId	NodeId	PropertyType	Mandatory
Conformance Units					
Auditing Connections					

This *EventType* inherits all *Properties* of the *AuditSecurityEventType*. Their semantic is defined in 6.4.4.

If the *Event* is generated by a *TransferSubscriptions Service* call, the *SourceNode Property* shall be assigned to the *SessionDiagnostics Object* that represents the session. The *SourceName* for *Events* of this type shall be "Session/TransferSubscriptions".

Otherwise, the *SourceNode Property* for *Events* of this type shall be assigned to the *Server Object*. The *SourceName* for *Events* of this type shall be "Session/" and the *Service* or cause that generates the *Event* (e.g. *CreateSession*, *ActivateSession* or *CloseSession*).

The *SessionId* shall contain the *SessionId* of the session that the *Service* call was issued on. In the *CreateSession Service* this shall be set to the newly created *SessionId*. If no session context exists (e.g. for a failed *CreateSession Service* call) the *SessionId* shall be null.

6.4.8 AuditCreateSessionEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 30.

Table 30 – AuditCreateSessionEventType definition

Attribute		Value			
BrowseName		AuditCreateSessionEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditSessionEventType</i> defined in 6.4.7, which means it inherits the InstanceDeclarations of that Node.					
HasProperty	Variable	SecureChannelId	String	PropertyType	Mandatory
HasProperty	Variable	ClientCertificate	ByteString	PropertyType	Mandatory
HasProperty	Variable	ClientCertificateThumbprint	String	PropertyType	Mandatory
HasProperty	Variable	RevisedSessionTimeout	Duration	PropertyType	Mandatory
Conformance Units					
Auditing Connections					

This *EventType* inherits all *Properties* of the *AuditSessionEventType*. Their semantic is defined in 6.4.7. The *SourceName* for *Events* of this type shall be "Session/CreateSession". The *ClientUserId* is not available for this call thus this parameter shall be set to the "System/CreateSession".

The additional *Properties* defined for this *EventType* reflect parameters of the *Service* call that triggers the *Event*.

SecureChannelId shall uniquely identify the SecureChannel. The application shall use the same identifier in all *AuditEvents* related to the Session Service Set (*AuditCreateSessionEventType*, *AuditActivateSessionEventType* and their subtypes) and the SecureChannel Service Set (*AuditChannelEventType* and its subtypes).

ClientCertificate is the clientCertificate parameter of the CreateSession Service call.

ClientCertificateThumbprint is a thumbprint of the *ClientCertificate*. See IEC 62541-6 for details on thumbprints.

RevisedSessionTimeout is the returned revisedSessionTimeout parameter of the CreateSession Service call.

6.4.9 AuditUrlMismatchEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 31.

Table 31 – AuditUrlMismatchEventType definition

Attribute		Value			
BrowseName		AuditUrlMismatchEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the AuditCreateSessionEventType defined in 6.4.8 which means it inherits the InstanceDeclarations of that Node.					
HasProperty	Variable	EndpointUrl	String	PropertyType	Mandatory
Conformance Units					
Auditing Connections					

This *EventType* inherits all *Properties* of the *AuditSessionEventType*. Their semantic is defined in 6.4.8.

The additional *Properties* defined for this *EventType* reflect parameters of the *Service* call that triggers the *Event*.

EndpointUrl is the *endpointUrl* parameter of the *CreateSession Service* call.

6.4.10 AuditActivateSessionEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 32.

Table 32 – AuditActivateSessionEventType definition

Attribute		Value			
BrowseName		AuditActivateSessionEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the <i>AuditSessionEventType</i> defined in 6.4.7, which means it inherits the InstanceDeclarations of that Node.					
HasProperty	Variable	ClientSoftwareCertificates	SignedSoftwareCertificate[]	PropertyType	Mandatory
HasProperty	Variable	UserIdentityToken	UserIdentityToken	PropertyType	Mandatory
HasProperty	Variable	SecureChannelId	String	PropertyType	Mandatory
Conformance Units					
Auditing Connections					

This *EventType* inherits all *Properties* of the *AuditSessionEventType*. Their semantic is defined in 6.4.7. The *SourceName* for *Events* of this type shall be "Session/ActivateSession".

The additional *Properties* defined for this *EventType* reflect parameters of the *Service* call that triggers the *Event*.

ClientSoftwareCertificates is the *clientSoftwareCertificates* parameter of the *ActivateSession Service* call.

UserIdentityToken reflects the *userIdentityToken* parameter of the *ActivateSession Service* call. For Username/Password tokens the password shall not be included.

SecureChannelId shall uniquely identify the *SecureChannel*. The application shall use the same identifier in all *AuditEvents* related to the Session Service Set (*AuditCreateSessionEventType*, *AuditActivateSessionEventType* and their subtypes) and the *SecureChannel Service* Set (*AuditChannelEventType* and its subtypes).

6.4.11 AuditCancelEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 33.

Table 33 – AuditCancelEventType definition

Attribute		Value			
BrowseName		AuditCancelEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditSessionEventType</i> defined in 6.4.7, i.e. inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	RequestHandle	UInt32	PropertyType	Mandatory
Conformance Units					
Auditing Connections					

This *EventType* inherits all *Properties* of the *AuditSessionEventType*. Their semantic is defined in 6.4.7. The *SourceName* for *Events* of this type shall be "Session/Cancel".

The additional *Properties* defined for this *EventType* reflect parameters of the *Service* call that triggers the *Event*.

RequestHandle is the requestHandle parameter of the Cancel *Service* call.

6.4.12 AuditCertificateEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 34.

Table 34 – AuditCertificateEventType definition

Attribute		Value			
BrowseName		AuditCertificateEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the <i>AuditSecurityEventType</i> defined in 6.4.7, which means it inherits the InstanceDeclarations of that Node.					
HasProperty	Variable	Certificate	ByteString	PropertyType	Mandatory
Conformance Units					
Auditing Connections					

This *EventType* inherits all *Properties* of the *AuditSecurityEventType*. Their semantic is defined in 6.4.4. The *SourceName* for *Events* of this type shall be "Security/Certificate".

Certificate is any certificate validated by a *Server* that encountered a validation issue (i.e. users, applications, etc.). Additional subtypes of this *EventType* will be defined representing the individual validation errors. This certificate can be matched to the *Service* that passed it (Session or SecureChannel Service Set) since the *AuditEvents* for these *Services* also included the Certificate.

6.4.13 AuditCertificateDataMismatchEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 35.

Table 35 – AuditCertificateDataMismatchEventType definition

Attribute		Value			
BrowseName		AuditCertificateDataMismatchEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditCertificateEventType</i> defined in 6.4.12, i.e. inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	InvalidHostname	String	PropertyType	Mandatory
HasProperty	Variable	InvalidUri	String	PropertyType	Mandatory
Conformance Units					
Auditing Connections					