



BSI Standards Publication

OPC Unified Architecture

Part 5: Information Model

National foreword

This British Standard is the UK implementation of EN IEC 62541-5:2020. It is identical to IEC 62541-5:2020. It supersedes [BS EN 62541-5:2015](#), which will be withdrawn on 4 March 2021.

The UK participation in its preparation was entrusted to Technical Committee GEL/65/3, Industrial communications: process measurement and control, including fieldbus.

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

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

© The British Standards Institution 2020
Published by BSI Standards Limited 2020

ISBN 978 0 580 51134 9

ICS 25.040.40; 25.100.01; 35.100.05

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

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 September 2020.

Amendments/corrigenda issued since publication

Date	Text affected
------	---------------

English Version

OPC Unified Architecture - Part 5: Information Model
(IEC 62541-5:2020)

Architecture unifiée OPC - Partie 5: Modèle d'information
(IEC 62541-5:2020)

OPC Unified Architecture - Teil 5: Informationsmodell
(IEC 62541-5:2020)

This European Standard was approved by CENELEC on 2020-08-14. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the CEN-CENELEC Management Centre or to any CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Bulgaria, Croatia, Cyprus, the Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Norway, Poland, Portugal, Republic of North Macedonia, Romania, Serbia, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.



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

CEN-CENELEC Management Centre: Rue de la Science 23, B-1040 Brussels

European foreword

The text of document 65E/717/FDIS, future edition 3 of IEC 62541-5, prepared by SC 65E "Devices and integration in enterprise systems" of IEC/TC 65 "Industrial-process measurement, control and automation" was submitted to the IEC-CENELEC parallel vote and approved by CENELEC as EN IEC 62541-5:2020.

The following dates are fixed:

- latest date by which the document has to be implemented at national level by publication of an identical national standard or by endorsement (dop) 2021-05-14
- latest date by which the national standards conflicting with the document have to be withdrawn (dow) 2023-08-14

This document supersedes EN 62541-5:2015 and all of its amendments and corrigenda (if any).

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

This document has been prepared under a mandate given to CENELEC by the European Commission and the European Free Trade Association.

Endorsement notice

The text of the International Standard IEC 62541-5:2020 was approved by CENELEC as a European Standard without any modification.

Annex ZA (normative)

Normative references to international publications with their corresponding European publications

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

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

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

<u>Publication</u>	<u>Year</u>	<u>Title</u>	<u>EN/HD</u>	<u>Year</u>
IEC/TR 62541-1	-	OPC unified architecture - Part 1: Overview and concepts	CLC/TR 62541-1	-
IEC 62541-3	-	OPC Unified Architecture - Part 3: Address Space Model	EN IEC 62541-3	-
IEC 62541-4	-	OPC Unified Architecture - Part 4: Services	EN IEC 62541-4	-
IEC 62541-6	-	OPC Unified Architecture - Part 6: Mappings	EN IEC 62541-6	-
IEC 62541-7	-	OPC unified architecture - Part 7: Profiles	EN IEC 62541-7	-
IEC 62541-9	-	OPC Unified Architecture - Part 9: Alarms and Conditions	EN IEC 62541-9	-
IEC 62541-10	-	OPC Unified Architecture - Part 10: Programs	EN IEC 62541-10	-
IEC 62541-11	-	OPC Unified Architecture - Part 11: Historical Access	EN IEC 62541-11	-
ISO/IEC/IEEE 60559	2011	Information technology - Microprocessor Systems - Floating-Point arithmetic	-	-
IETF RFC 2045	-	Multipurpose Internet Mail Extensions (MIME) Part 1: Format of Internet Message Bodies	-	-
IETF RFC 2046	-	Multipurpose Internet Mail Extensions (MIME) - Part Two: Media Types	-	-

EN IEC 62541-5:2020 (E)

<u>Publication</u>	<u>Year</u>	<u>Title</u>	<u>EN/HD</u>	<u>Year</u>
IETF RFC 2047	-	Multipurpose Internet Mail Extensions - Part Three: Message Header Extensions for Non-ASCII Text	-	-
XML-1	-	XML Schema Part 1: Structures, W3C	-	-
XML-2	-	XML Schema Part 2: Datatypes, W3C	-	-
W3C Xpath	-	XML Path Language (XPath)	-	-
IETF RFC 3629	-	UTF-8, a transformation format of ISO 10646	-	-

CONTENTS

FOREWORD	14
1 Scope	16
2 Normative references	16
3 Terms, definitions, abbreviated terms and conventions	17
3.1 Terms and definitions	17
3.2 Abbreviated terms	17
3.3 Conventions for Node descriptions	17
4 NodeIds and BrowseNames	19
4.1 NodeIds	19
4.2 BrowseNames	19
5 Common Attributes	19
5.1 General	19
5.2 Objects	20
5.3 Variables	20
5.4 VariableTypes	21
5.5 Methods	21
6 Standard ObjectTypes	21
6.1 General	21
6.2 BaseObjectType	22
6.3 ObjectTypes for the Server Object	22
6.3.1 ServerType	22
6.3.2 ServerCapabilitiesType	25
6.3.3 ServerDiagnosticsType	28
6.3.4 SessionsDiagnosticsSummaryType	29
6.3.5 SessionDiagnosticsObjectType	29
6.3.6 VendorServerInfoType	30
6.3.7 ServerRedundancyType	30
6.3.8 TransparentRedundancyType	31
6.3.9 NonTransparentRedundancyType	31
6.3.10 NonTransparentNetworkRedundancyType	32
6.3.11 OperationLimitsType	33
6.3.12 AddressSpaceFileType	34
6.3.13 NamespaceMetadataType	35
6.3.14 NamespacesType	36
6.4 ObjectTypes used as EventTypes	37
6.4.1 General	37
6.4.2 BaseEventType	37
6.4.3 AuditEventType	40
6.4.4 AuditSecurityEventType	41
6.4.5 AuditChannelEventType	42
6.4.6 AuditOpenSecureChannelEventType	42
6.4.7 AuditSessionEventType	43
6.4.8 AuditCreateSessionEventType	44
6.4.9 AuditUriMismatchEventType	45
6.4.10 AuditActivateSessionEventType	45

6.4.11	AuditCancelEventType	46
6.4.12	AuditCertificateEventType	47
6.4.13	AuditCertificateDataMismatchEventType	47
6.4.14	AuditCertificateExpiredEventType	48
6.4.15	AuditCertificateInvalidEventType	48
6.4.16	AuditCertificateUntrustedEventType	48
6.4.17	AuditCertificateRevokedEventType	49
6.4.18	AuditCertificateMismatchEventType	49
6.4.19	AuditNodeManagementEventType	50
6.4.20	AuditAddNodesEventType	50
6.4.21	AuditDeleteNodesEventType	50
6.4.22	AuditAddReferencesEventType	51
6.4.23	AuditDeleteReferencesEventType	51
6.4.24	AuditUpdateEventType	52
6.4.25	AuditWriteUpdateEventType	52
6.4.26	AuditHistoryUpdateEventType	53
6.4.27	AuditUpdateMethodEventType	54
6.4.28	SystemEventType	54
6.4.29	DeviceFailureEventType	54
6.4.30	SystemStatusChangeEvent	55
6.4.31	BaseModelChangeEvent	55
6.4.32	GeneralModelChangeEvent	56
6.4.33	SemanticChangeEvent	56
6.4.34	EventQueueOverflowEventType	57
6.4.35	ProgressEventType	57
6.5	ModellingRuleType	58
6.6	FolderType	58
6.7	DataTypeEncodingType	58
6.8	AggregateFunctionType	58
7	Standard VariableTypes	59
7.1	General	59
7.2	BaseVariableType	59
7.3	PropertyType	59
7.4	BaseDataVariableType	60
7.5	ServerVendorCapabilityType	60
7.6	ServerStatusType	61
7.7	BuildInfoType	61
7.8	ServerDiagnosticsSummaryType	62
7.9	SamplingIntervalDiagnosticsArrayType	63
7.10	SamplingIntervalDiagnosticsType	63
7.11	SubscriptionDiagnosticsArrayType	63
7.12	SubscriptionDiagnosticsType	64
7.13	SessionDiagnosticsArrayType	65
7.14	SessionDiagnosticsVariableType	66
7.15	SessionSecurityDiagnosticsArrayType	69
7.16	SessionSecurityDiagnosticsType	69
7.17	OptionSetType	70
7.18	SelectionListType	71
7.19	AudioVariableType	72

8	Standard Objects and their Variables.....	73
8.1	General.....	73
8.2	Objects used to organise the AddressSpace structure	73
8.2.1	Overview	73
8.2.2	Root	73
8.2.3	Views	74
8.2.4	Objects	74
8.2.5	Types	75
8.2.6	ObjectTypes	75
8.2.7	VariableTypes	76
8.2.8	ReferenceTypes	77
8.2.9	DataTypes	78
8.2.10	EventTypes	78
8.3	Server Object and its containing Objects.....	79
8.3.1	General	79
8.3.2	Server Object	80
8.4	ModellingRule Objects	80
8.4.1	ExposesItsArray	80
8.4.2	Mandatory	81
8.4.3	Optional.....	81
8.4.4	OptionalPlaceholder	81
8.4.5	MandatoryPlaceholder	82
9	Standard Methods	82
9.1	GetMonitoredItems	82
9.2	ResendData.....	83
9.3	SetSubscriptionDurable	83
9.4	RequestServerStateChange.....	84
10	Standard Views	85
11	Standard ReferenceTypes	85
11.1	References	85
11.2	HierarchicalReferences.....	86
11.3	NonHierarchicalReferences	86
11.4	HasChild.....	87
11.5	Aggregates	87
11.6	Organizes	87
11.7	HasComponent	87
11.8	HasOrderedComponent	88
11.9	HasProperty.....	88
11.10	HasSubtype	88
11.11	HasModellingRule.....	89
11.12	HasTypeDefinition.....	89
11.13	HasEncoding	89
11.14	HasEventSource	90
11.15	HasNotifier.....	90
11.16	GeneratesEvent.....	90
11.17	AlwaysGeneratesEvent	91
12	Standard DataTypes.....	91
12.1	Overview.....	91

12.2	DataTypes defined in IEC 62541-3.....	91
12.3	DataTypes defined in IEC 62541-4.....	98
12.4	BuildInfo	99
12.5	RedundancySupport	99
12.6	ServerState.....	100
12.7	RedundantServerDataType	101
12.8	SamplingIntervalDiagnosticsDataType	101
12.9	ServerDiagnosticsSummaryDataType	102
12.10	ServerStatusDataType	102
12.11	SessionDiagnosticsDataType.....	103
12.12	SessionSecurityDiagnosticsDataType	106
12.13	ServiceCounterDataType	107
12.14	StatusResult	107
12.15	SubscriptionDiagnosticsDataType.....	107
12.16	ModelChangeStructureDataType	109
12.17	SemanticChangeStructureDataType	110
12.18	BitFieldMaskDataType	110
12.19	NetworkGroupDataType.....	110
12.20	EndpointUrlListDataType	111
12.21	KeyValuePair	111
12.22	EndpointType.....	112
Annex A	(informative) Design decisions when modelling the server information	113
A.1	Overview.....	113
A.2	ServerType and Server Object	113
A.3	Typed complex Objects beneath the Server Object	113
A.4	Properties versus DataVariables	113
A.5	Complex Variables using complex DataTypes	114
A.6	Complex Variables having an array.....	114
A.7	Redundant information.....	114
A.8	Usage of the BaseDataVariableType.....	115
A.9	Subtyping	115
A.10	Extensibility mechanism.....	115
Annex B	(normative) StateMachines	116
B.1	General.....	116
B.2	Examples of finite state machines	116
B.2.1	Simple state machine	116
B.2.2	State machine containing substates	117
B.3	Definition of state machine.....	118
B.4	Representation of state machines in the AddressSpace	118
B.4.1	Overview	118
B.4.2	StateMachineType	119
B.4.3	StateVariableType	120
B.4.4	TransitionVariableType.....	121
B.4.5	FiniteStateMachineType	122
B.4.6	FiniteStateVariableType	124
B.4.7	FiniteTransitionVariableType	125
B.4.8	StateType	126
B.4.9	InitialStateType	126
B.4.10	TransitionType.....	127

B.4.11	FromState.....	127
B.4.12	ToState.....	128
B.4.13	HasCause.....	128
B.4.14	HasEffect.....	129
B.4.15	HasSubStateMachine	129
B.4.16	TransitionEventType.....	130
B.4.17	AuditUpdateStateEventType	130
B.4.18	Special Restrictions on subtyping StateMachines	131
B.4.19	Specific StatusCodes for StateMachines.....	131
B.5	Examples of StateMachines in the AddressSpace	132
B.5.1	StateMachineType using inheritance	132
B.5.2	StateMachineType with a SubStateMachine using inheritance	133
B.5.3	StateMachineType using containment.....	134
B.5.4	Example of a StateMachine having Transition to SubStateMachine	134
Annex C	(normative) File Transfer	137
C.1	Overview.....	137
C.2	FileType.....	137
C.2.1	General	137
C.2.2	Open	138
C.2.3	Close.....	139
C.2.4	Read	140
C.2.5	Write.....	141
C.2.6	GetPosition.....	142
C.2.7	SetPosition	142
C.3	File System.....	143
C.3.1	FileDirectoryType	143
C.3.2	FileSystem Object	144
C.3.3	CreateDirectory	144
C.3.4	CreateFile.....	145
C.3.5	Delete.....	146
C.3.6	MoveOrCopy	147
C.4	Temporary file transfer.....	148
C.4.1	TemporaryFileTransferType.....	148
C.4.2	File transfer sequences	149
C.4.3	GenerateFileForRead	150
C.4.4	GenerateFileForWrite	151
C.4.5	CloseAndCommit.....	151
C.4.6	FileTransferStateMachineType	152
C.4.7	Reset.....	155
Annex D	(normative) DataTypeDictionary.....	156
D.1	Overview.....	156
D.2	Data Type Model.....	156
D.3	DataTypeDictionary, DataTypeDescription, DataTypeEncoding and DataTypeSystem.....	157
D.4	AddressSpace organization.....	159
D.5	Node definitions	161
D.5.1	HasDescription	161
D.5.2	DataTypeDictionaryType.....	161
D.5.3	DataTypeDescriptionType.....	162

D.5.4	DataTypeSystemType	162
D.5.5	OPC Binary	162
D.5.6	XML Schema	163
Annex E (normative)	OPC Binary Type Description System	164
E.1	Concepts	164
E.2	Schema description	165
E.2.1	TypeDictionary	165
E.2.2	TypeDescription	165
E.2.3	OpaqueType	166
E.2.4	EnumeratedType	166
E.2.5	StructuredType	167
E.2.6	FieldType	167
E.2.7	EnumeratedValue	169
E.2.8	ByteOrder	169
E.2.9	ImportDirective	170
E.3	Standard Type descriptions	170
E.4	Type description examples	171
E.4.1	A 128-bit signed integer	171
E.4.2	A 16-bit value divided into several fields	171
E.4.3	A structured type with optional fields	171
E.4.4	An array of integers	171
E.4.5	An array of integers with a terminator instead of a length prefix	171
E.4.6	A simple union	171
E.4.7	An enumerated type	172
E.4.8	A nillable array	172
E.5	OPC Binary XML schema	172
E.6	OPC Binary Standard TypeDictionary	174
Annex F (normative)	User Authorization	176
F.1	Overview	176
F.2	RoleSetType	176
F.2.1	RoleSetType definition	176
F.2.2	AddRole Method	176
F.2.3	RemoveRole Method	177
F.3	RoleType	178
F.3.1	RoleType definition	178
F.3.2	IdentityMappingRuleType	179
F.3.3	AddIdentity Method	180
F.3.4	RemoveIdentity Method	180
F.3.5	AddApplication Method	181
F.3.6	RemoveApplication Method	181
F.3.7	AddEndpoint Method	182
F.3.8	RemoveEndpoint Method	182
F.4	RoleMappingRuleChangedAuditEventType	183
Figure 1 – Standard AddressSpace structure		73
Figure 2 – Views organization		74
Figure 3 – Objects organization		75
Figure 4 – ObjectTypes organization		76

Figure 5 – VariableTypes organization	76
Figure 6 – ReferenceType definitions	77
Figure 7 – EventTypes organization	78
Figure 8 – Excerpt of diagnostic information of the Server	80
Figure B.1 – Example of a simple state machine	117
Figure B.2 – Example of a state machine having a sub-machine	117
Figure B.3 – The StateMachine Information Model	119
Figure B.4 – Example of a FiniteStateMachine type	124
Figure B.5 – Example of a FiniteStateMachine instance	124
Figure B.6 – Example of an initial State in a sub-machine	126
Figure B.7 – Example of a StateMachineType using inheritance	132
Figure B.8 – Example of a StateMachineType with a SubStateMachine using inheritance	133
Figure B.9 – Example of a StateMachineType using containment	134
Figure B.10 – Example of a StateMachine with Transitions from sub-states	135
Figure B.11 – Example of a StateMachineType having Transition to SubStateMachine	136
Figure C.1 – FileSystem example	144
Figure C.2 – Read file transfer example sequence	149
Figure C.3 – Write file transfer example sequence	149
Figure C.4 – File transfer States	152
Figure C.5 – FileTransferStateMachineType	153
Figure D.1 – DataType model	156
Figure D.2 – Example of DataType modelling	159
Figure D.3 – DataTypes organization	160
Figure E.1 – OPC Binary Dictionary structure	164
Table 1 – Examples of DataTypes	18
Table 2 – TypeDefinitionTable	18
Table 3 – Common Node Attributes	20
Table 4 – Common Object Attributes	20
Table 5 – Common Variable Attributes	21
Table 6 – Common VariableType Attributes	21
Table 7 – Common Method Attributes	21
Table 8 – BaseObjectType definition	22
Table 9 – ServerType definition	23
Table 10 – ServerCapabilitiesType definition	26
Table 11 – ServerDiagnosticsType definition	28
Table 12 – SessionsDiagnosticsSummaryType definition	29
Table 13 – SessionDiagnosticsObjectType definition	30
Table 14 – VendorServerInfoType definition	30
Table 15 – ServerRedundancyType definition	31
Table 16 – TransparentRedundancyType definition	31
Table 17 – NonTransparentRedundancyType definition	32

Table 18 – NonTransparentNetworkRedundancyType definition	32
Table 19 – OperationLimitsType definition	33
Table 20 – AddressSpaceFileType definition.....	34
Table 21 – NamespaceMetadataType definition	35
Table 22 – NamespacesType definition.....	37
Table 23 – BaseEventType definition	38
Table 24 – AuditEventType definition	41
Table 25 – AuditSecurityEventType definition	42
Table 26 – AuditChannelEventType definition	42
Table 27 – AuditOpenSecureChannelEventType definition.....	43
Table 28 – AuditSessionEventType definition.....	44
Table 29 – AuditCreateSessionEventType definition	44
Table 30 – AuditUrlMismatchEventType definition.....	45
Table 31 – AuditActivateSessionEventType definition	46
Table 32 – AuditCancelEventType definition	46
Table 33 – AuditCertificateEventType definition	47
Table 34 – AuditCertificateDataMismatchEventType definition	47
Table 35 – AuditCertificateExpiredEventType definition	48
Table 36 – AuditCertificateInvalidEventType definition.....	48
Table 37 – AuditCertificateUntrustedEventType definition	49
Table 38 – AuditCertificateRevokedEventType definition	49
Table 39 – AuditCertificateMismatchEventType definition	49
Table 40 – AuditNodeManagementEventType definition.....	50
Table 41 – AuditAddNodesEventType definition	50
Table 42 – AuditDeleteNodesEventType definition	51
Table 43 – AuditAddReferencesEventType definition	51
Table 44 – AuditDeleteReferencesEventType definition	52
Table 45 – AuditUpdateEventType definition.....	52
Table 46 – AuditWriteUpdateEventType definition.....	53
Table 47 – AuditHistoryUpdateEventType definition	53
Table 48 – AuditUpdateMethodEventType definition	54
Table 49 – SystemEventType definition	54
Table 50 – DeviceFailureEventType definition.....	55
Table 51 – SystemStatusChangeEvent definition	55
Table 52 – BaseModelChangeEvent definition.....	55
Table 53 – GeneralModelChangeEvent definition	56
Table 54 – SemanticChangeEvent definition	56
Table 55 – EventQueueOverflowEventType definition	57
Table 56 – ProgressEventType definition	57
Table 57 – ModellingRuleType definition.....	58
Table 58 – FolderType definition.....	58
Table 59 – DataTypeEncodingType definition	58
Table 60 – AggregateFunctionType definition	59

Table 61 – BaseVariableType definition	59
Table 62 – PropertyType definition	60
Table 63 – BaseDataVariableType definition.....	60
Table 64 – ServerVendorCapabilityType definition	61
Table 65 – ServerStatusType definition.....	61
Table 66 – BuildInfoType definition	62
Table 67 – ServerDiagnosticsSummaryType definition.....	62
Table 68 – SamplingIntervalDiagnosticsArrayType definition	63
Table 69 – SamplingIntervalDiagnosticsType definition.....	63
Table 70 – SubscriptionDiagnosticsArrayType definition	64
Table 71 – SubscriptionDiagnosticsType definition	65
Table 72 – SessionDiagnosticsArrayType definition	66
Table 73 – SessionDiagnosticsVariableType definition.....	67
Table 74 – SessionSecurityDiagnosticsArrayType definition	69
Table 75 – SessionSecurityDiagnosticsType definition.....	70
Table 76 – OptionSetType definition	71
Table 77 – SelectionListType definition.....	72
Table 78 – AudioVariableType definition	72
Table 79 – Root definition	73
Table 80 – Views definition	74
Table 81 – Objects definition	75
Table 82 – Types definition	75
Table 83 – ObjectTypes definition	76
Table 84 – VariableTypes definition	77
Table 85 – ReferenceTypes definition	78
Table 86 – DataTypes definition.....	78
Table 87 – EventTypes definition	79
Table 88 – Server definition	80
Table 89 – ExposesItsArray definition	81
Table 90 – Mandatory definition	81
Table 91 – Optional definition	81
Table 92 – OptionalPlaceholder definition	81
Table 93 – MandatoryPlaceholder definition.....	82
Table 94 – GetMonitoredItems Method AddressSpace definition	83
Table 95 – ResendData Method AddressSpace definition	83
Table 96 – SetSubscriptionDurable Method AddressSpace definition	84
Table 97 – RequestServerStateChange Method AddressSpace definition	85
Table 98 – References ReferenceType	86
Table 99 – HierarchicalReferences ReferenceType.....	86
Table 100 – NonHierarchicalReferences ReferenceType	86
Table 101 – HasChild ReferenceType.....	87
Table 102 – Aggregates ReferenceType	87
Table 103 – Organizes ReferenceType	87

Table 104 – HasComponent ReferenceType	88
Table 105 – HasOrderedComponent ReferenceType	88
Table 106 – HasProperty ReferenceType	88
Table 107 – HasSubtype ReferenceType	89
Table 108 – HasModellingRule ReferenceType	89
Table 109 – HasTypeDefinition ReferenceType	89
Table 110 – HasEncoding ReferenceType	90
Table 111 – HasEventSource ReferenceType	90
Table 112 – HasNotifier ReferenceType	90
Table 113 – GeneratesEvent ReferenceType	91
Table 114 – AlwaysGeneratesEvent ReferenceType	91
Table 115 – IEC 62541-3 DataType definitions	92
Table 116 – BaseDataType definition	93
Table 117 – Structure definition	94
Table 118 – Enumeration definition	95
Table 119 – ByteString definition	95
Table 120 – Number definition	95
Table 121 – Double definition	96
Table 122 – Integer definition	96
Table 123 – DateTime definition	96
Table 124 – String definition	96
Table 125 – UInteger definition	97
Table 126 – Image definition	97
Table 127 – UInt64 definition	97
Table 128 – DataTypeDefinition definition	97
Table 129 – EnumValueType definition	98
Table 130 – IEC 62541-4 DataType definitions	98
Table 131 – UserIdentityToken definition	99
Table 132 – BuildInfo structure	99
Table 133 – BuildInfo definition	99
Table 134 – RedundancySupport values	100
Table 135 – RedundancySupport definition	100
Table 136 – ServerState values	100
Table 137 – ServerState definition	101
Table 138 – RedundantServerDataType Structure	101
Table 139 – RedundantServerDataType definition	101
Table 140 – SamplingIntervalDiagnosticsDataType Structure	101
Table 141 – SamplingIntervalDiagnosticsDataType definition	101
Table 142 – ServerDiagnosticsSummaryDataType Structure	102
Table 143 – ServerDiagnosticsSummaryDataType definition	102
Table 144 – ServerStatusDataType Structure	103
Table 145 – ServerStatusDataType definition	103
Table 146 – SessionDiagnosticsDataType Structure	104

Table 147 – SessionDiagnosticsDataType definition	106
Table 148 – SessionSecurityDiagnosticsDataType Structure	106
Table 149 – SessionSecurityDiagnosticsDataType definition.....	106
Table 150 – ServiceCounterDataType Structure	107
Table 151 – ServiceCounterDataType definition.....	107
Table 152 – StatusResult Structure.....	107
Table 153 – StatusResult definition.....	107
Table 154 – SubscriptionDiagnosticsDataType structure	108
Table 155 – SubscriptionDiagnosticsDataType definition	109
Table 156 – ModelChangeStructureDataType structure	109
Table 157 – ModelChangeStructureDataType definition	109
Table 158 – SemanticChangeStructureDataType structure	110
Table 159 – SemanticChangeStructureDataType definition	110
Table 160 – BitFieldMaskDataType definition.....	110
Table 161 – NetworkGroupDataType Structure	111
Table 162 – NetworkGroupDataType definition	111
Table 163 – EndpointUrlListDataType Structure.....	111
Table 164 – EndpointUrlListDataType definition	111
Table 165 – KeyValuePair structure	111
Table 166 – EndpointType structure.....	112
Table B.1 – StateMachineType definition	120
Table B.2 – StateVariableType definition	121
Table B.3 – TransitionVariableType definition	122
Table B.4 – FiniteStateMachineType definition	123
Table B.5 – FiniteStateVariableType definition.....	125
Table B.6 – FiniteTransitionVariableType definition	125
Table B.7 – StateType definition	126
Table B.8 – InitialStateType definition.....	127
Table B.9 – TransitionType definition	127
Table B.10 – FromState ReferenceType	128
Table B.11 – ToState ReferenceType	128
Table B.12 – HasCause ReferenceType.....	129
Table B.13 – HasEffect ReferenceType	129
Table B.14 – HasSubStateMachine ReferenceType	130
Table B.15 – TransitionEventType	130
Table B.16 – AuditUpdateStateEventType	131
Table B.17 – Specific StatusCodes for StateMachines	131
Table C.1 – FileType.....	138
Table C.2 – Open Method AddressSpace definition	139
Table C.3 – Close Method AddressSpace definition	140
Table C.4 – Read Method AddressSpace definition.....	141
Table C.5 – Write Method AddressSpace definition.....	142
Table C.6 – GetPosition Method AddressSpace definition.....	142

Table C.7 – SetPosition Method AddressSpace definition	143
Table C.8 – FileDirectoryType.....	143
Table C.9 – CreateDirectory Method AddressSpace definition	145
Table C.10 – CreateFile Method AddressSpace definition.....	146
Table C.11 – Delete Method AddressSpace definition	147
Table C.12 – MoveOrCopy Method AddressSpace definition.....	148
Table C.13 – TemporaryFileTransferType	148
Table C.14 – GenerateFileForRead Method AddressSpace definition	150
Table C.15 – GenerateFileForWrite Method AddressSpace definition.....	151
Table C.16 – CloseAndCommit Method AddressSpace definition	152
Table C.17 – FileTransferStateMachineType.....	154
Table C.18 – FileTransferStateMachineType transitions	155
Table D.1 – HasDescription ReferenceType.....	161
Table D.2 – DataTypeDictionaryType definition.....	161
Table D.3 – DataTypeDescriptionType definition.....	162
Table D.4 – DataTypeSystemType definition.....	162
Table D.5 – OPC Binary definition	162
Table D.6 – XML Schema definition	163
Table E.1 – TypeDictionary components	165
Table E.2 – TypeDescription components	166
Table E.3 – OpaqueType components	166
Table E.4 – EnumeratedType components.....	167
Table E.5 – StructuredType components.....	167
Table E.6 – FieldType components	168
Table E.7 – EnumeratedValue components.....	169
Table E.8 – ImportDirective components.....	170
Table E.9 – Standard Type descriptions.....	170
Table F.1 – RoleSetType definition	176
Table F.2 – RoleType definition	178
Table F.3 – IdentityMappingRuleType	179
Table F.4 – RoleMappingRuleChangedAuditEventType definition	183

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 third edition cancels and replaces the second edition published in 2015. This edition constitutes a technical revision.

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

- a) Added Annex F on User Authentication. Describes the Role Information Model that also allows configuration of Roles.
- b) Added new data types: "Union", "Decimal", "OptionSet", "DateString", "TimeString", "DurationString", "NormalizedString", "DecimalString", and "AudioDataType".
- c) Added Method to request a state change in a Server.
- d) Added Method to set Subscription to persistent mode.

- e) Added Method to request resending of data from a Subscription.
- f) Added concept allowing to temporarily create a file to write to or read from a server in C.4.
- g) Added new Variable type to support Selection Lists.
- h) Added optional properties to FiniteStateMachineType to expose currently available states and transitions.
- i) Added UrisVersion Property to ServerType. This version information can be used for session-less service invocation.

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

FDIS	Report on voting
65E/717/FDIS	65E/733/RVD

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 Clause 3 in one of the parts of the 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 also often 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 –

Part 5: Information Model

1 Scope

This part of IEC 62541 defines the Information Model of the OPC Unified Architecture. The Information Model describes standardized *Nodes* of a *Server's AddressSpace*. These *Nodes* are standardized types as well as standardized 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) applies.

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

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

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

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

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

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*

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

IETF RFC 2045, Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies
<http://www.ietf.org/rfc/rfc2045.txt>

IETF RFC 2046, Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types
<https://www.ietf.org/rfc/rfc2046.txt>

IETF RFC 2047, Multipurpose Internet Mail Extensions (MIME) Part Three: Message Header Extensions for Non-ASCII Text
<http://www.ietf.org/rfc/rfc2047.txt>

XML Schema Part 1: Structures
<http://www.w3.org/TR/xmlschema-1/>

XML Schema Part 2: Datatypes
<http://www.w3.org/TR/xmlschema-2/>

Xpath: XML Path Language
<http://www.w3.org/TR/xpath/>

IETF RFC 3629: UTF-8, a transformation format of ISO 10646
<http://www.ietf.org/rfc/rfc3629.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 TR 62541-1 and IEC 62541-3 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at <http://www.electropedia.org/>
- ISO Online browsing platform: available at <http://www.iso.org/obp>

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. 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.

Table 2 –TypeDefinitionTable

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
<i>ReferenceType</i> name	<i>NodeClass</i> of the <i>TargetNode</i> .	<i>BrowseName</i> of the target <i>Node</i> . If the <i>Reference</i> is to be instantiated by the server, then the value of the target <i>Node</i> 's <i>BrowseName</i> is "--".	<i>DataType</i> of the referenced <i>Node</i> , only applicable for <i>Variables</i> .	<i>TypeDefinition</i> of the referenced <i>Node</i> , only applicable for <i>Variables</i> and <i>Objects</i> .	Referenced <i>ModellingRule</i> of the referenced <i>Object</i> .
NOTE Notes referencing footnotes of the table content.					

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 document are only symbolic names. IEC 62541-6 defines the actual *NodeIds*.

The symbolic name of each *Node* defined in this document 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 document, 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 document is 0. The namespace for this NamespaceIndex is specified in IEC 62541-3.

Note that this document 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 document 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 this document, 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 document.
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 depends 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 this document, 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 this document, 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 document, 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 in the table defining the <i>Variable</i> .
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 this document, 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 in the table defining the <i>VariableType</i> .

5.5 Methods

For all *Methods* specified in this document, 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 document 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 document 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 document. 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*.

In addition to the *ObjectTypes* in Clause 6, Annex B provides *ObjectTypes* for StateMachines, Annex C provides *ObjectTypes* Model for File Transfer and Annex F defines *ObjectTypes* for User Authorization.

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* except for *HasSubtype 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	ModellingRule	
HasSubtype	ObjectType	ServerType	Defined in 6.3.1			
HasSubtype	ObjectType	ServerCapabilitiesType	Defined in 6.3.2			
HasSubtype	ObjectType	ServerDiagnosticsType	Defined in 6.3.3			
HasSubtype	ObjectType	SessionsDiagnosticsSummaryType	Defined in 6.3.4			
HasSubtype	ObjectType	SessionDiagnosticsObjectType	Defined in 6.3.5			
HasSubtype	ObjectType	VendorServerInfoType	Defined in 6.3.6			
HasSubtype	ObjectType	ServerRedundancyType	Defined in 6.3.7			
HasSubtype	ObjectType	BaseEventType	Defined in 6.4.2			
HasSubtype	ObjectType	ModellingRuleType	Defined in 6.5			
HasSubtype	ObjectType	FolderType	Defined in 6.6			
HasSubtype	ObjectType	DataTypeEncodingType	Defined in 6.7			

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.

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 ^a	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 ^a	-- ServerCapabilitiesType	Mandatory
HasComponent	Object	ServerDiagnostics ^a	-- ServerDiagnosticsType	Mandatory
HasComponent	Object	VendorServerInfo	-- VendorServerInfoType	Mandatory
HasComponent	Object	ServerRedundancy ^a	-- 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
^a Containing <i>Objects</i> and <i>Variables</i> of these <i>Objects</i> and <i>Variables</i> are defined by their <i>BrowseName</i> defined in the corresponding <i>TypeDefinitionNode</i> . The <i>NodeId</i> is defined by the composed symbolic name described in 4.1.				

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

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

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

NamespaceArray defines an array of namespace URIs. This *Variable* is also referred as *namespace table*. The indexes into the *namespace table* are referred to as *NamespaceIndexes*. *NamespaceIndexes* are used in *NodeIds* in OPC UA *Services*, rather than the longer 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. IEC 62541-4 defines required sub-ranges for different scenarios. The intent is to provide the clients an indication of availability among redundant *Servers*.

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*.

Namespaces provides a list of *NamespaceMetadataType* *Objects* with additional information about the namespaces used in the *Server*. See 6.3.13 for the description of *NamespaceMetadataType*.

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

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

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

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

6.3.2 ServerCapabilitiesType

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

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

ServerProfileArray lists the *Profiles* that the *Server* supports. See IEC 62541-7 for the definitions of *Server Profiles*. This list should be limited to the *Profiles* the *Server* supports in its current configuration.

LocaleIdArray is an array of *LocaleIds* that are known to be supported by the *Server*. The *Server* might not be aware of all *LocaleIds* that it supports because it may provide access to underlying servers, systems or devices that do not report the *LocaleIds* that they support.

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

MaxBrowseContinuationPoints is an integer specifying the maximum number of parallel continuation points of the Browse *Service* that the *Server* can support per session. The value specifies the maximum the *Server* can support under normal circumstances, so there is no

guarantee the *Server* can always support the maximum. The client should not open more Browse calls with open continuation points than exposed in this *Variable*. The value 0 indicates that the *Server* does not restrict the number of parallel continuation points the client should use.

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

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

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

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

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

The *MaxByteStringLength Property* indicates the maximum number of bytes in a *ByteString* supported by *Variables* of the *Server*. It also specifies the default maximum size of a *FileType Object's* read and write buffers. *Servers* may override this setting by adding the *MaxByteStringLength Property* defined in IEC 62541-3 to an individual *DataVariable* or *FileType Object*. If a *Server* does not impose a maximum number of bytes or is not able to determine the maximum number of bytes, this *Property* shall not be provided.

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

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

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

The *RoleSet Object* is used to publish all *Roles* supported by the *Server*. The *RoleSetType* is specified in F.2.

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

6.3.3 ServerDiagnosticsType

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

Table 11 – ServerDiagnosticsType definition

Attribute	Value			
BrowseName	ServerDiagnosticsType			
IsAbstract	False			
References	Node Class	BrowseName	Data Type / 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

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 at all, 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
NOTE This row represents no <i>Node</i> in the <i>AddressSpace</i> . It is a placeholder pointing out that instances of the <i>ObjectType</i> will have those <i>Objects</i> .				

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.

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	DataType / 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

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 shall 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					

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
HasSubtype	ObjectType	TransparentRedundancyType	Defined in 6.3.8		
HasSubtype	ObjectType	NonTransparentRedundancyType	Defined in 6.3.9		

RedundancySupport indicates what redundancy is supported by the *Server*. Its values are defined in 12.5. It shall be set to *NONE_0* for all instances of the *ServerRedundancyType* using the *ObjectType* directly (no subtype).

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

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 ServerRedundancyType defined in 6.3.7, which means it inherits the InstanceDeclarations of that Node.					
HasProperty	Variable	ServerUriArray	String[]	PropertyType	Mandatory
HasSubtype	ObjectType	NonTransparentNetworkRedundancyType	Defined in 6.3.10		

ServerUriArray is an array with the URI of all redundant *Servers* of the OPC UA *Server*. See IEC 62541-4 for the definition of redundancy in this document. In a non-transparent redundancy environment, the *Client* is responsible to subscribe to the redundant *Servers*. Therefore the *Client* might open a session to one or more redundant *Servers* of this array. The *ServerUriArray* shall contain the local *Server*.

RedundancySupport is inherited from the *ServerRedundancyType*. It shall be set to COLD_1, WARM_2, HOT_3 or HOT_AND_MIRRORED_5 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 NonTransparentRedundancyType defined in 6.3.9, which means it inherits the InstanceDeclarations of that Node.					
HasProperty	Variable	ServerNetworkGroups	NetworkGroupDataType[]	PropertyType	Mandatory

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_1, WARM_2, HOT_3 or HOT_AND_MIRRORED_5 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_5.

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

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 <i>InstanceDeclarations</i> 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

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 C.2					
HasComponent	Method	ExportNamespace	The method has no parameters.		Optional

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	UInt16	PropertyType	Optional

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*. 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.

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 – NamespaceType 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

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*.

6.4 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 23.

Table 23 – 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					
HasSubtype	ObjectType	AuditEventType	Defined in 6.4.3		
HasSubtype	ObjectType	SystemEventType	Defined in 6.4.28		
HasSubtype	ObjectType	BaseModelChangeEventType	Defined in 6.4.31		
HasSubtype	ObjectType	SemanticChangeEventType	Defined in 6.4.33		
HasSubtype	ObjectType	EventQueueOverflowEventType	Defined in 6.4.34		
HasSubtype	ObjectType	ProgressEventType	Defined in 6.4.35		
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

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

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

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

6.4.3 AuditEventType

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

Table 24 – 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.					
HasSubtype	ObjectType	AuditSecurityEventType	Defined in 6.4.4		
HasSubtype	ObjectType	AuditNodeManagementEventType	Defined in 6.4.19		
HasSubtype	ObjectType	AuditUpdateEventType	Defined in 6.4.24		
HasSubtype	ObjectType	AuditUpdateMethodEventType	Defined in 6.4.27		
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

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 62541-3.

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.

6.4.4 AuditSecurityEventType

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

Table 25 – 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.					
HasSubtype	ObjectType	AuditChannelEventType	Defined in 6.4.5		
HasSubtype	ObjectType	AuditSessionEventType	Defined in 6.4.7		
HasSubtype	ObjectType	AuditCertificateEventType	Defined in 6.4.12		
HasProperty	Variable	StatusCodeId	StatusCode	PropertyType	Optional

This *EventType* inherits all *Properties* of the *AuditEventType*. Their semantic is defined in 6.4.3. There are no additional *Properties* defined for this *EventType*.

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 26.

Table 26 – 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.					
HasSubtype	ObjectType	AuditOpenSecureChannelEventType	Defined in 6.4.6		
HasProperty	Variable	SecureChannelId	String	PropertyType	Mandatory

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 27.

Table 27 – AuditOpenSecureChannelEventType definition

Attribute		Value			
BrowseName		AuditOpenSecureChannelEventType			
IsAbstract		True			
References	Node Class	BrowseName	DataType	TypeDefinition	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

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.

6.4.7 AuditSessionEventType

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

Table 28 – AuditSessionEventType definition

Attribute		Value			
BrowseName		AuditSessionEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditSecurityEventType</i> defined in 6.4.4, which means it inherits the InstanceDeclarations of that Node.					
HasSubtype	ObjectType	AuditCreateSessionEventType	Defined in 6.4.8		
HasSubtype	ObjectType	AuditActivateSessionEventType	Defined in 6.4.10		
HasSubtype	ObjectType	AuditCancelEventType	Defined in 6.4.11		
HasProperty	Variable	SessionId	NodeId	PropertyType	Mandatory

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 29.

Table 29 – 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.					
HasSubtype	ObjectType	AuditUrlMismatchEventType	Defined in 6.4.9		
HasProperty	Variable	SecureChannelId	String	PropertyType	Mandatory
HasProperty	Variable	ClientCertificate	ByteString	PropertyType	Mandatory
HasProperty	Variable	ClientCertificateThumbprint	String	PropertyType	Mandatory
HasProperty	Variable	RevisedSessionTimeout	Duration	PropertyType	Mandatory

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 30.

Table 30 – AuditUrlMismatchEventType definition

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

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 31.