

INTERNATIONAL STANDARD

NORME INTERNATIONALE



**OPC unified architecture –
Part 5: Information Model**

**Architecture unifiée OPC –
Partie 5: Modèle d'informations**

Withdrawn



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2015 IEC, Geneva, Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either IEC or IEC's member National Committee in the country of the requester. If you have any questions about IEC copyright or have an enquiry about obtaining additional rights to this publication, please contact the address below or your local IEC member National Committee for further information.

Droits de reproduction réservés. Sauf indication contraire, aucune partie de cette publication ne peut être reproduite ni utilisée sous quelque forme que ce soit et par aucun procédé, électronique ou mécanique, y compris la photocopie et les microfilms, sans l'accord écrit de l'IEC ou du Comité national de l'IEC du pays du demandeur. Si vous avez des questions sur le copyright de l'IEC ou si vous désirez obtenir des droits supplémentaires sur cette publication, utilisez les coordonnées ci-après ou contactez le Comité national de l'IEC de votre pays de résidence.

IEC Central Office
3, rue de Varembe
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00
info@iec.ch
www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

IEC Catalogue - webstore.iec.ch/catalogue

The stand-alone application for consulting the entire bibliographical information on IEC International Standards, Technical Specifications, Technical Reports and other documents. Available for PC, Mac OS, Android Tablets and iPad.

IEC publications search - www.iec.ch/searchpub

The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee,...). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished

Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and also once a month by email.

Electropedia - www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 30 000 terms and definitions in English and French, with equivalent terms in 15 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary

More than 60 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

IEC Customer Service Centre - webstore.iec.ch/csc

If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: csc@iec.ch.

A propos de l'IEC

La Commission Electrotechnique Internationale (IEC) est la première organisation mondiale qui élabore et publie des Normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications IEC

Le contenu technique des publications IEC est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

Catalogue IEC - webstore.iec.ch/catalogue

Application autonome pour consulter tous les renseignements bibliographiques sur les Normes internationales, Spécifications techniques, Rapports techniques et autres documents de l'IEC. Disponible pour PC, Mac OS, tablettes Android et iPad.

Recherche de publications IEC - www.iec.ch/searchpub

La recherche avancée permet de trouver des publications IEC en utilisant différents critères (numéro de référence, texte, comité d'études,...). Elle donne aussi des informations sur les projets et les publications remplacées ou retirées.

IEC Just Published - webstore.iec.ch/justpublished

Restez informé sur les nouvelles publications IEC. Just Published détaille les nouvelles publications parues. Disponible en ligne et aussi une fois par mois par email.

Electropedia - www.electropedia.org

Le premier dictionnaire en ligne de termes électroniques et électriques. Il contient plus de 30 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans 15 langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (IEV) en ligne.

Glossaire IEC - std.iec.ch/glossary

Plus de 60 000 entrées terminologiques électrotechniques, en anglais et en français, extraites des articles Termes et Définitions des publications IEC parues depuis 2002. Plus certaines entrées antérieures extraites des publications des CE 37, 77, 86 et CISPR de l'IEC.

Service Clients - webstore.iec.ch/csc

Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions contactez-nous: csc@iec.ch.

INTERNATIONAL STANDARD

NORME INTERNATIONALE



**OPC unified architecture –
Part 5: Information Model**

**Architecture unifiée OPC –
Partie 5: Modèle d'informations**

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

COMMISSION
ELECTROTECHNIQUE
INTERNATIONALE

ICS 25.040.40; 35.100

ISBN 978-2-8322-2384-0

**Warning! Make sure that you obtained this publication from an authorized distributor.
Attention! Veuillez vous assurer que vous avez obtenu cette publication via un distributeur agréé.**

CONTENTS

FOREWORD	12
1 Scope	14
2 Normative references	14
3 Terms, definitions and conventions	14
3.1 Terms and definitions	14
3.2 Abbreviations and symbols	14
3.3 Conventions for Node descriptions	15
4 NodeIds and BrowseNames	16
4.1 NodeIds	16
4.2 BrowseNames	16
5 Common Attributes	17
5.1 General	17
5.2 Objects	17
5.3 Variables	17
5.4 VariableTypes	17
6 Standard ObjectTypes	18
6.1 General	18
6.2 BaseObjectType	18
6.3 ObjectTypes for the Server Object	18
6.3.1 ServerType	18
6.3.2 ServerCapabilitiesType	20
6.3.3 ServerDiagnosticsType	22
6.3.4 SessionsDiagnosticsSummaryType	23
6.3.5 SessionDiagnosticsObjectType	24
6.3.6 VendorServerInfoType	25
6.3.7 ServerRedundancyType	25
6.3.8 TransparentRedundancyType	25
6.3.9 NonTransparentRedundancyType	26
6.3.10 NonTransparentNetworkRedundancyType	26
6.3.11 OperationLimitsType	27
6.3.12 AddressSpaceFileType	29
6.3.13 NamespaceMetadataType	29
6.3.14 NamespacesType	31
6.4 ObjectTypes used as EventTypes	31
6.4.1 General	31
6.4.2 BaseEventType	31
6.4.3 AuditEventType	33
6.4.4 AuditSecurityEventType	34
6.4.5 AuditChannelEventType	35
6.4.6 AuditOpenSecureChannelEventType	35
6.4.7 AuditSessionEventType	36
6.4.8 AuditCreateSessionEventType	36
6.4.9 AuditUrlMismatchEventType	37
6.4.10 AuditActivateSessionEventType	38
6.4.11 AuditCancelEventType	38
6.4.12 AuditCertificateEventType	39

6.4.13	AuditCertificateDataMismatchEventType.....	39
6.4.14	AuditCertificateExpiredEventType.....	39
6.4.15	AuditCertificateInvalidEventType.....	40
6.4.16	AuditCertificateUntrustedEventType.....	40
6.4.17	AuditCertificateRevokedEventType.....	40
6.4.18	AuditCertificateMismatchEventType.....	41
6.4.19	AuditNodeManagementEventType.....	41
6.4.20	AuditAddNodesEventType.....	42
6.4.21	AuditDeleteNodesEventType.....	42
6.4.22	AuditAddReferencesEventType.....	42
6.4.23	AuditDeleteReferencesEventType.....	43
6.4.24	AuditUpdateEventType.....	43
6.4.25	AuditWriteUpdateEventType.....	44
6.4.26	AuditHistoryUpdateEventType.....	44
6.4.27	AuditUpdateMethodEventType.....	45
6.4.28	SystemEventType.....	45
6.4.29	DeviceFailureEventType.....	45
6.4.30	SystemStatusChangeEventType.....	46
6.4.31	BaseModelChangeEventType.....	46
6.4.32	GeneralModelChangeEventType.....	46
6.4.33	SemanticChangeEventType.....	47
6.4.34	EventQueueOverflowEventType.....	47
6.4.35	ProgressEventType.....	48
6.5	ModellingRuleType.....	48
6.6	FolderType.....	48
6.7	DataTypeEncodingType.....	49
6.8	DataTypeSystemType.....	49
6.9	AggregateFunctionType.....	49
7	Standard VariableTypes.....	50
7.1	General.....	50
7.2	BaseVariableType.....	50
7.3	PropertyType.....	50
7.4	BaseDataVariableType.....	50
7.5	ServerVendorCapabilityType.....	51
7.6	DataTypeDictionaryType.....	51
7.7	DataTypeDescriptionType.....	52
7.8	ServerStatusType.....	52
7.9	BuildInfoType.....	52
7.10	ServerDiagnosticsSummaryType.....	53
7.11	SamplingIntervalDiagnosticsArrayType.....	53
7.12	SamplingIntervalDiagnosticsType.....	54
7.13	SubscriptionDiagnosticsArrayType.....	54
7.14	SubscriptionDiagnosticsType.....	54
7.15	SessionDiagnosticsArrayType.....	55
7.16	SessionDiagnosticsVariableType.....	56
7.17	SessionSecurityDiagnosticsArrayType.....	57
7.18	SessionSecurityDiagnosticsType.....	58
7.19	OptionSetType.....	58
8	Standard Objects and their Variables.....	59

8.1	General.....	59
8.2	Objects used to organise the AddressSpace structure	59
8.2.1	Overview	59
8.2.2	Root.....	60
8.2.3	Views.....	60
8.2.4	Objects	61
8.2.5	Types	61
8.2.6	ObjectTypes	62
8.2.7	VariableTypes.....	63
8.2.8	ReferenceTypes.....	64
8.2.9	DataTypes	64
8.2.10	OPC Binary.....	66
8.2.11	XML Schema	66
8.2.12	EventTypes.....	66
8.3	Server Object and its containing Objects.....	67
8.3.1	General.....	67
8.3.2	Server Object.....	68
8.4	ModellingRule Objects	69
8.4.1	ExposesItsArray.....	69
8.4.2	Mandatory.....	69
8.4.3	Optional.....	69
8.4.4	OptionalPlaceholder.....	70
8.4.5	MandatoryPlaceholder.....	70
9	Standard Methods	70
9.1	GetMonitoredItems	70
10	Standard Views	71
11	Standard ReferenceTypes	71
11.1	References	71
11.2	HierarchicalReferences	71
11.3	NonHierarchicalReferences	71
11.4	HasChild	72
11.5	Aggregates	72
11.6	Organizes	72
11.7	HasComponent	73
11.8	HasOrderedComponent	73
11.9	HasProperty.....	73
11.10	HasSubtype	73
11.11	HasModellingRule.....	74
11.12	HasTypeDefinition.....	74
11.13	HasEncoding	74
11.14	HasDescription	75
11.15	HasEventSource	75
11.16	HasNotifier.....	75
11.17	GeneratesEvent	75
11.18	AlwaysGeneratesEvent	76
12	Standard DataTypes	76
12.1	Overview.....	76
12.2	DataTypes defined in IEC 62541-3.....	76

12.3	DataTypes defined in IEC 62541-4.....	81
12.4	BuildInfo	82
12.5	RedundancySupport	82
12.6	ServerState.....	83
12.7	RedundantServerDataType	83
12.8	SamplingIntervalDiagnosticsDataType	84
12.9	ServerDiagnosticsSummaryDataType	84
12.10	ServerStatusDataType	85
12.11	SessionDiagnosticsDataType.....	86
12.12	SessionSecurityDiagnosticsDataType	87
12.13	ServiceCounterDataType	88
12.14	StatusResult	88
12.15	SubscriptionDiagnosticsDataType.....	89
12.16	ModelChangeStructureDataType	90
12.17	SemanticChangeStructureDataType	90
12.18	BitFieldMaskDataType	91
12.19	NetworkGroupDataType.....	91
12.20	EndpointUrlListDataType	92
Annex A	(informative) Design decisions when modelling the server information	93
A.1	Overview.....	93
A.2	ServerType and Server Object	93
A.3	Typed complex Objects beneath the Server Object	93
A.4	Properties versus DataVariables	93
A.5	Complex Variables using complex DataTypes	94
A.6	Complex Variables having an array.....	94
A.7	Redundant information.....	94
A.8	Usage of the BaseDataVariableType	95
A.9	Subtyping	95
A.10	Extensibility mechanism.....	95
Annex B	(normative) StateMachines	96
B.1	General.....	96
B.2	Examples of finite state machines	96
B.2.1	Simple state machine.....	96
B.2.2	State machine containing substates	97
B.3	Definition of state machine.....	98
B.4	Representation of state machines in the AddressSpace	98
B.4.1	Overview	98
B.4.2	StateMachineType	99
B.4.3	StateVariableType	100
B.4.4	TransitionVariableType	101
B.4.5	FiniteStateMachineType	101
B.4.6	FiniteStateVariableType.....	102
B.4.7	FiniteTransitionVariableType	103
B.4.8	StateType	103
B.4.9	InitialStateType.....	104
B.4.10	TransitionType	105
B.4.11	FromState.....	105
B.4.12	ToState.....	106
B.4.13	HasCause	106

B.4.14	HasEffect	106
B.4.15	HasSubStateMachine	107
B.4.16	TransitionEventType	107
B.4.17	AuditUpdateStateEventType	108
B.4.18	Special Restrictions on subtyping StateMachines	108
B.4.19	Specific StatusCodes for StateMachines	109
B.5	Examples of StateMachines in the AddressSpace	110
B.5.1	StateMachineType using inheritance	110
B.5.2	StateMachineType with a sub-machine using inheritance	111
B.5.3	StateMachineType using containment	112
B.5.4	Example of a StateMachine having Transition to SubStateMachine	113
Annex C (normative)	File Transfer	115
C.1	Overview	115
C.2	FileType	115
C.3	Open	116
C.4	Close	117
C.5	Read	117
C.6	Write	118
C.7	GetPosition	118
C.8	SetPosition	119
Figure 1	Standard AddressSpace Structure	59
Figure 2	Views Organization	60
Figure 3	Objects Organization	61
Figure 4	ObjectTypes Organization	62
Figure 5	VariableTypes Organization	63
Figure 6	ReferenceType Definitions	64
Figure 7	DataTypes Organization	65
Figure 8	EventTypes Organization	67
Figure 9	Excerpt of Diagnostic Information of the Server	68
Figure B.1	Example of a simple state machine	97
Figure B.2	Example of a state machine having a sub-machine	97
Figure B.3	The StateMachine Information Model	99
Figure B.4	Example of an initial State in a sub-machine	104
Figure B.5	Example of a StateMachineType using inheritance	110
Figure B.6	Example of a StateMachineType with a SubStateMachine using inheritance	111
Figure B.7	Example of a StateMachineType using containment	112
Figure B.8	Example of a state machine with transitions from sub-states	113
Figure B.9	Example of a StateMachineType having Transition to SubStateMachine	114
Table 1	Examples of DataTypes	15
Table 2	Type Definition Table	16
Table 3	Common Node Attributes	17
Table 4	Common Object Attributes	17
Table 5	Common Variable Attributes	17

Table 6 – Common VariableType Attributes	18
Table 7 – BaseObjectType Definition	18
Table 8 – ServerType Definition.....	19
Table 9 – ServerCapabilitiesType Definition.....	21
Table 10 – ServerDiagnosticsType Definition	23
Table 11 – SessionsDiagnosticsSummaryType Definition	24
Table 12 – SessionDiagnosticsObjectType Definition	24
Table 13 – VendorServerInfoType Definition	25
Table 14 – ServerRedundancyType Definition.....	25
Table 15 – TransparentRedundancyType Definition	25
Table 16 – NonTransparentRedundancyType Definition	26
Table 17 – NonTransparentNetworkRedundancyType Definition	27
Table 18 – OperationLimitsType Definition.....	28
Table 19 – AddressSpaceFileType Definition	29
Table 20 – NamespaceMetadataType Definition	30
Table 21 – NamespacesType Definition	31
Table 22 – BaseEventType Definition	31
Table 23 – AuditEventType Definition.....	34
Table 24 – AuditSecurityEventType Definition.....	34
Table 25 – AuditChannelEventType Definition	35
Table 26 – AuditOpenSecureChannelEventType Definition	35
Table 27 – AuditSessionEventType Definition.....	36
Table 28 – AuditCreateSessionEventType Definition.....	37
Table 29 – AuditUriMismatchEventType Definition	37
Table 30 – AuditActivateSessionEventType Definition.....	38
Table 31 – AuditCancelEventType Definition.....	38
Table 32 – AuditCertificateEventType Definition	39
Table 33 – AuditCertificateDataMismatchEventType Definition	39
Table 34 – AuditCertificateExpiredEventType Definition.....	40
Table 35 – AuditCertificateInvalidEventType Definition	40
Table 36 – AuditCertificateUntrustedEventType Definition	40
Table 37 – AuditCertificateRevokedEventType Definition	41
Table 38 – AuditCertificateMismatchEventType Definition.....	41
Table 39 – AuditNodeManagementEventType Definition	41
Table 40 – AuditAddNodesEventType Definition	42
Table 41 – AuditDeleteNodesEventType Definition	42
Table 42 – AuditAddReferencesEventType Definition.....	43
Table 43 – AuditDeleteReferencesEventType Definition.....	43
Table 44 – AuditUpdateEventType Definition	43
Table 45 – AuditWriteUpdateEventType Definition	44
Table 46 – AuditHistoryUpdateEventType Definition	44
Table 47 – AuditUpdateMethodEventType Definition.....	45
Table 48 – SystemEventType Definition.....	45

Table 49 – DeviceFailureEventType Definition	46
Table 50 – SystemStatusChangeEventDefinition	46
Table 51 – BaseModelChangeEventDefinition	46
Table 52 – GeneralModelChangeEventDefinition	47
Table 53 – SemanticChangeEventDefinition	47
Table 54 – EventQueueOverflowEventType Definition	47
Table 55 – ProgressEventType Definition	48
Table 56 – ModellingRuleType Definition	48
Table 57 – FolderType Definition	49
Table 58 – DataTypeEncodingType Definition	49
Table 59 – DataTypeSystemType Definition	49
Table 60 – AggregateFunctionType Definition	49
Table 61 – BaseVariableType Definition	50
Table 62 – PropertyType Definition	50
Table 63 – BaseDataVariableType Definition	51
Table 64 – ServerVendorCapabilityType Definition	51
Table 65 – DataTypeDictionaryType Definition	51
Table 66 – DataTypeDescriptionType Definition	52
Table 67 – ServerStatusType Definition	52
Table 68 – BuildInfoType Definition	53
Table 69 – ServerDiagnosticsSummaryType Definition	53
Table 70 – SamplingIntervalDiagnosticsArrayType Definition	54
Table 71 – SamplingIntervalDiagnosticsType Definition	54
Table 72 – SubscriptionDiagnosticsArrayType Definition	54
Table 73 – SubscriptionDiagnosticsType Definition	55
Table 74 – SessionDiagnosticsArrayType Definition	55
Table 75 – SessionDiagnosticsVariableType Definition	56
Table 76 – SessionSecurityDiagnosticsArrayType Definition	58
Table 77 – SessionSecurityDiagnosticsType Definition	58
Table 78 – OptionSetType Definition	59
Table 79 – Root Definition	60
Table 80 – Views Definition	61
Table 81 – Objects Definition	61
Table 82 – Types Definition	62
Table 83 – ObjectTypes Definition	63
Table 84 – VariableTypes Definition	63
Table 85 – ReferenceTypes Definition	64
Table 86 – DataTypes Definition	66
Table 87 – OPC Binary Definition	66
Table 88 – XML Schema Definition	66
Table 89 – EventTypes Definition	67
Table 90 – Server Definition	69
Table 91 – ExposesItsArray Definition	69

Table 92 – Mandatory Definition	69
Table 93 – Optional Definition	70
Table 94 – OptionalPlaceholder Definition	70
Table 95 – MandatoryPlaceholder Definition	70
Table 96 – GetMonitoredItems Method AddressSpace Definition	71
Table 97 – References ReferenceType	71
Table 98 – HierarchicalReferences ReferenceType	71
Table 99 – NonHierarchicalReferences ReferenceType	72
Table 100 – HasChild ReferenceType	72
Table 101 – Aggregates ReferenceType	72
Table 102 – Organizes ReferenceType	73
Table 103 – HasComponent ReferenceType	73
Table 104 – HasOrderedComponent ReferenceType	73
Table 105 – HasProperty ReferenceType	73
Table 106 – HasSubtype ReferenceType	74
Table 107 – HasModellingRule ReferenceType	74
Table 108 – HasTypeDefinition ReferenceType	74
Table 109 – HasEncoding ReferenceType	74
Table 110 – HasDescription ReferenceType	75
Table 111 – HasEventSource ReferenceType	75
Table 112 – HasNotifier ReferenceType	75
Table 113 – GeneratesEvent ReferenceType	76
Table 114 – AlwaysGeneratesEvent ReferenceType	76
Table 115 – IEC 62541-3 DataType Definitions	77
Table 116 – BaseDataType Definition	78
Table 117 – Structure Definition	78
Table 118 – Enumeration Definition	79
Table 119 – ByteString Definition	79
Table 120 – Number Definition	79
Table 121 – Double Definition	79
Table 122 – Integer Definition	80
Table 123 – DateTime Definition	80
Table 124 – String Definition	80
Table 125 – UInteger Definition	80
Table 126 – Image Definition	80
Table 127 – UInt64 Definition	81
Table 128 – IEC 62541-4 DataType Definitions	81
Table 129 – UserIdentityToken Definition	82
Table 130 – BuildInfo Structure	82
Table 131 – BuildInfo Definition	82
Table 132 – RedundancySupport Values	82
Table 133 – RedundancySupport Definition	83
Table 134 – ServerState Values	83

Table 135 – ServerState Definition	83
Table 136 – RedundantServerDataType Structure	83
Table 137 – RedundantServerDataType Definition	84
Table 138 – SamplingIntervalDiagnosticsDataType Structure	84
Table 139 – SamplingIntervalDiagnosticsDataType Definition	84
Table 140 – ServerDiagnosticsSummaryDataType Structure	85
Table 141 – ServerDiagnosticsSummaryDataType Definition	85
Table 142 – ServerStatusDataType Structure	85
Table 143 – ServerStatusDataType Definition	86
Table 144 – SessionDiagnosticsDataType Structure	86
Table 145 – SessionDiagnosticsDataType Definition	87
Table 146 – SessionSecurityDiagnosticsDataType Structure	88
Table 147 – SessionSecurityDiagnosticsDataType Definition	88
Table 148 – ServiceCounterDataType Structure	88
Table 149 – ServiceCounterDataType Definition	88
Table 150 – StatusResult Structure	89
Table 151 – StatusResult Definition	89
Table 152 – SubscriptionDiagnosticsDataType Structure	89
Table 153 – SubscriptionDiagnosticsDataType Definition	90
Table 154 – ModelChangeStructureDataType Structure	90
Table 155 – ModelChangeStructureDataType Definition	90
Table 156 – SemanticChangeStructureDataType Structure	91
Table 157 – SemanticChangeStructureDataType Definition	91
Table 158 – BitFieldMaskDataType Definition	91
Table 159 – NetworkGroupDataType Structure	91
Table 160 – NetworkGroupDataType Definition	91
Table 161 – EndpointUrlListDataType Structure	92
Table 162 – EndpointUrlListDataType Definition	92
Table B.1 – StateMachineType Definition	100
Table B.2 – StateVariableType Definition	100
Table B.3 – TransitionVariableType Definition	101
Table B.4 – FiniteStateMachineType Definition	102
Table B.5 – FiniteStateVariableType Definition	103
Table B.6 – FiniteTransitionVariableType Definition	103
Table B.7 – StateType Definition	104
Table B.8 – InitialStateType Definition	105
Table B.9 – TransitionType Definition	105
Table B.10 – FromState ReferenceType	105
Table B.11 – ToState ReferenceType	106
Table B.12 – HasCause ReferenceType	106
Table B.13 – HasEffect ReferenceType	107
Table B.14 – HasSubStateMachine ReferenceType	107
Table B.15 – TransitionEventType	107

Table B.16 – AuditUpdateStateEventType	108
Table B.17 – Specific StatusCodes for StateMachines	109
Table C.1 – FileType.....	115
Table C.2 – Open Method AddressSpace Definition	117
Table C.3 – Close Method AddressSpace Definition	117
Table C.4 – Read Method AddressSpace Definition	118
Table C.5 – Write Method AddressSpace Definition	118
Table C.6 – GetPosition Method AddressSpace Definition	119
Table C.7 – SetPosition Method AddressSpace Definition.....	119

Withdrawn

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

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

- a) Defined `ProgressEventType` in 6.4.35 identifying the progress of an operation such as a service call (issue number 0057);
- b) Defined `DataType` called `BitFieldMaskDataType` in 12.18 representing a bit field where individual fields can be written without redefining other fields (issue number 0188);
- c) Delete Property `SamplingRateCount` in `ServerDiagnosticSummaryDataType` (12.9) as it was not needed (issue number 0635);

- d) Added the Property “EffectiveTransitionTime” to TransitionVariableType in B.4.4 (issue number 0728);
- e) Introduced VariableType OptionSetType in 7.19 representing a bit mask and text defining the semantic of the individual bits (issue number 0983);
- f) Added a new EventType called SystemStatusChangeEvent in 6.4.30 that can be used to indicate connection to the underlying system is lost (issue number 1416);
- g) Added properties to ServerCapabilitiesType (6.3.2) describing the max array length and string length for variables as well as added an object for operation limits (max size of arrays when calling services (e.g. read)). Added type OperationLimitsType (6.3.11) containing that information (issue number 1451);
- h) Added SecureChannelId to AuditActivateSessionEventType (6.4.10) and adapted text in various places (issue number 1492);
- i) Added normative Annex C defining FileType and Methods used to transfer files (issue number 1502);
- j) Added a Method GetMonitoredItems on ServerType (6.3.1) to receive information on monitored items (issue 1543);
- k) Removed the concept of *ModelParent* from document as it is not that useful. The *NodeId* of the *ReferenceType* will be kept not breaking existing applications (issue numbers 1555 and 1556).
- l) Added meta data for namespaces in ServerType (6.3.1) and created types for managing that (issue number 1702).
- m) Added representations for ModellingRules OptionalPlaceholder in 8.4.4 and MandatoryPlaceholder in 8.4.5 (issue number 1831);
- n) Added new types NonTransparentNetworkRedundancyType (6.3.10), NetworkGroupDataType (12.19) and EndpointUrlListDataType (12.20) to manage HotAndMirrored redundancy. Added more description on redundancy and updated RedundancySupport enumeration in 12.5 (issue number 2031);

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

CDV	Report on voting
65E/376/CDV	65E/404/RVC

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

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

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

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

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

IMPORTANT – The “colour inside” logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this publication 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 standardised *Nodes* of a *Server's AddressSpace*. These *Nodes* are standardised types as well as standardised instances used for diagnostics or as entry points to server-specific *Nodes*. Thus, the Information Model defines the *AddressSpace* of an empty OPC UA *Server*. However, it is not expected that all *Servers* will provide all of these *Nodes*.

2 Normative references

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

3 Terms, definitions 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, as well as the following apply.

3.1.1

ClientUserId

string that identifies the user of the client requesting an action

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

3.2 Abbreviations and symbols

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 – Type Definition Table

Attribute	Value				
Attribute name	Attribute value. If it is an optional Attribute that is not set “--“ will be used.				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
<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>Attributes</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 standard are only symbolic names. IEC 62541-6 defines the actual *NodeIds*.

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

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

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

4.2 BrowseNames

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

5 Common Attributes

5.1 General

For all *Nodes* specified in this standard, 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 vendor specific.
Description	Optionally a vendor 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 <i>Attributes</i> to not writeable that are not said to be vendor-specific. For example, the <i>Description Attribute</i> may be set to writeable since a <i>Server</i> may provide a server-specific description for the <i>Node</i> . The <i>NodeId</i> shall not be writeable, because it is defined for each <i>Node</i> in this standard.
UserWriteMask	Optionally the <i>UserWriteMask Attribute</i> can be provided. The same rules as for the <i>WriteMask Attribute</i> apply.

5.2 Objects

For all *Objects* specified in this standard, 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 vendor specific.

5.3 Variables

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

Table 5 – Common Variable Attributes

Attribute	Value
MinimumSamplingInterval	Optionally, a vendor-specific minimum sampling interval is provided.
AccessLevel	The access level for <i>Variables</i> used for type definitions is vendor-specific, for all other <i>Variables</i> defined in this standard, the access level shall allow a current read; other settings are vendor specific.
UserAccessLevel	The value for the <i>UserAccessLevel Attribute</i> is vendor-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 vendor-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 vendor-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> .

5.4 VariableTypes

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

Table 6 – Common VariableType Attributes

Attributes	Value
Value	Optionally a vendor-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 vendor-specific. If the <i>ValueRank</i> specifies an array of a specific dimension (i.e. <i>ValueRank</i> > 0) then the <i>ArrayDimensions</i> <i>Attribute</i> shall be specified in the table defining the <i>VariableType</i> .

6 Standard ObjectTypes

6.1 General

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

6.2 BaseObjectType

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

There are no *References* except for *HasSubtype* *References* specified for this *ObjectType*. It is formally defined in Table 7.

Table 7 – BaseObjectType Definition

Attribute	Value				
BrowseName	BaseObjectType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
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		
HasSubtype	ObjectType	DataTypeSystemType	Defined in 6.8		

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 8.

Table 8 – 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
HasComponent	Variable	ServerStatus ¹	ServerStatusDataType	ServerStatusType	Mandatory
HasProperty	Variable	ServiceLevel	Byte	PropertyType	Mandatory
HasProperty	Variable	Auditing	Boolean	PropertyType	Mandatory
HasComponent	Object	ServerCapabilities ¹	-	ServerCapabilitiesType	Mandatory
HasComponent	Object	ServerDiagnostics ¹	-	ServerDiagnosticsType	Mandatory
HasComponent	Object	VendorServerInfo	-	VendorServerInfoType	Mandatory
HasComponent	Object	ServerRedundancy ¹	-	ServerRedundancyType	Mandatory
HasComponent	Object	Namespaces	-	NamespacesType	Optional
HasComponent	Method	GetMonitoredItems	Defined in 9		Optional
NOTE 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.

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. The concrete

values are vendor-specific. 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*.

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 6.3.14 for the description of *NamespaceMetadataType*.

The *GetMonitoredItems* *Method* is used to identify the *MonitoredItems* of a subscription. It is defined in 9; 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 9.

Table 9 – 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
HasComponent	Object	OperationLimits	-- OperationLimitsType	Optional
HasComponent	Object	ModellingRules	-- FolderType	Mandatory
HasComponent	Object	AggregateFunctions	-- FolderType	Mandatory
HasComponent	Variable	Vendor specific Variables of a subtype of the ServerVendorCapabilityType defined in 7.5		--

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 length of Strings supported by Variables of the Server. The Server might further restrict the String length for individual Variables without notice to the client. Servers may use the *Property MaxStringLength* defined in IEC 62541-3 on individual *DataVariables* to specify the length on individual values. The individual *Property* may have larger or smaller values than *MaxStringLength*.

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 remaining components of the *ServerCapabilitiesType* define the server-specific capabilities of the Server. Each is defined using a *HasComponent Reference* whose target is an instance of a vendor-defined subtype of the abstract *ServerVendorCapabilityType* (see 7.5). Each subtype of this type defines a specific Server capability. The *NodeIds* for these Variables and their *VariableTypes* are server-defined.

6.3.3 ServerDiagnosticsType

This *ObjectType* defines diagnostic information about the OPC UA Server. This *ObjectType* is formally defined in Table 10.

Table 10 – ServerDiagnosticsType Definition

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

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 7.11.

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 7.13. 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 statistic information is collected, and setting the value to FALSE disables the collection without resetting the statistic values.

Static diagnostic *Nodes* that always appear in the *AddressSpace* will return *Bad_NotReadable* when the *Value Attribute* of such a *Node* is read or subscribed to and diagnostics are turned off. Dynamic diagnostic *Nodes* (such as the *Session Nodes*) will not appear in the *AddressSpace* when diagnostics are turned off.

6.3.4 SessionsDiagnosticsSummaryType

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

Table 11 – 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 12.

Table 12 – SessionDiagnosticsObjectType Definition

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

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 7.13.

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 13.

Table 13 – 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 14.

Table 14 – 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 15.

Table 15 – 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 16.

Table 16 – NonTransparentRedundancyType Definition

Attribute	Value				
BrowseName	NonTransparentRedundancyType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the <i>ServerRedundancyType</i> defined in 6.3.7, which means it inherits the <i>InstanceDeclarations</i> of that Node.					
HasProperty	Variable	ServerUriArray	String[]	PropertyType	Mandatory
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 standard. 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*. The *Client* is allowed to access the redundant *Server* only as described there, however, "hot" switchover implies the support of "warm" switchover and "warm" switchover implies the support of "cold" switchover. Support for HotAndMirrored redundancy implies the support of "hot" switchover, however, for *Servers* supporting HotAndMirrored redundancy it is strongly recommended that *Clients* use the HotAndMirrored mechanisms.

If the *Server* supports only a "cold" switchover, the *ServiceLevel Variable* of the *Server Object* should be considered to identify the primary *Server*. In this scenario, only the primary *Server* may be able to access the underlying system, because the underlying system may support access only from a single *Server*. In this case, all other *Servers* will be identified with a *ServiceLevel* of zero.

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 17.

Table 17 – 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 ones provided in the *ServerUriArray*. However, the order might be different. Thus the array represents a list of HotAndMirrored redundant *Servers*. If a server only supports network redundancy, it has only one entry in the *ServerNetworkGroups*. The *networkPaths* in the structure represents the redundant network paths for each of the *Servers*. The *networkPaths* describes the different paths (one entry for each path) ordered by priority. Each network path contains an *endpointUriList* 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 18.

Table 18 – OperationLimitsType Definition

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

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 using historyReadDetails RAW, PROCESSED, MODIFIED or ATTIME.

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

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*, and the maximum size of the *linksToAdd* and the *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 19. It represents an XML address space file using the XML schema defined in IEC 62541-6.

Table 19 – 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 20.

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 *StaticNodeIdsTypes*, *StaticNumericNodeIdRange* and *StaticStringNodeIdPattern Properties*.

Table 20 – 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	StaticNodeIdsTypes	IdType[]	PropertyType	Mandatory
HasProperty	Variable	StaticNumericNodeIdRange	NumericRange[]	PropertyType	Mandatory
HasProperty	Variable	StaticStringNodeIdPattern	String	PropertyType	Mandatory
HasComponent	Object	NamespaceFile	-	AddressSpaceFileType	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.

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*.

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.

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 *StaticNodeIdsTypes 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 *StaticNodeIdsTypes 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 *StaticNodeIdsTypes 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 XML schema is defined in IEC 62541-6. The XML file is provided through an *AddressSpaceFileType Object*.

6.3.14 NamespacesType

This *ObjectType* defines a list of *NamespaceMetadataType Objects* provided by the *Server*. It is formally defined in Table 21.

Table 21 – NamespacesType Definition

Attribute	Value				
BrowseName	NamespacesType				
IsAbstract	False				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modelling Rule
Subtype of the <i>BaseObjectType</i> defined in 6.2					
HasComponent	Object	<NamespaceIdentifier>	-	NamespaceMetadataType	Optional Placeholder

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*.

6.4 ObjectTypes used as EventTypes

6.4.1 General

This International Standard 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 22.

Table 22 – BaseEventType Definition

Attribute	Value				
BrowseName	BaseEventType				
IsAbstract	True				
References	NodeClass	BrowseName	Data Type	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	BaseModelChangeEvent	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.

SourceNode 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 *SourceNode*.

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 – 1 000
MEDIUM HIGH	601 – 800
MEDIUM	401 – 600
MEDIUM LOW	201 – 400
LOW	1 – 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 – 1 000)	15	1 000
	14	955
	13	910
	12	865
	11	820
	10	775
	9	730
	8	685
MEDIUM (334 – 666)	7	650
	6	575
	5	500
	4	425
	3	350
	2	300
LOW (1 – 333)	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 23.

Table 23 – 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* should 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 24.

Table 24 – AuditSecurityEventType Definition

Attribute		Value			
BrowseName		AuditSecurityEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditEventType</i> 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		

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*.

6.4.5 AuditChannelEventType

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

Table 25 – AuditChannelEventType Definition

Attribute	Value				
BrowseName	AuditChannelEventType				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the <i>AuditSecurityEventType</i> defined in 6.4.4, which means it inherits the <i>InstanceDeclarations</i> 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. There are no additional *Properties* defined for this *EventType*. The *SourceNode* for *Events* of this type should be assigned to the *Server Object*. The *SourceName* for *Events* of this type should 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 26.

Table 26 – 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 <i>InstanceDeclarations</i> 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 should 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 27.

Table 27 – AuditSessionEventType Definition

Attribute		Value			
BrowseName		AuditSessionEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditEventType</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 *AuditEventType*. Their semantic is defined in 6.4.4.

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

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

The *SessionId* should 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* is set to null.

6.4.8 AuditCreateSessionEventType

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

Table 28 – 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 should 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 29.

Table 29 – 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 InstanceDeclarations 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 30.

Table 30 – AuditActivateSessionEventType Definition

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

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

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

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

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

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

6.4.11 AuditCancelEventType

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

Table 31 – AuditCancelEventType Definition

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

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

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

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

6.4.12 AuditCertificateEventType

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

Table 32 – AuditCertificateEventType Definition

Attribute		Value			
BrowseName		AuditCertificateEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the <i>AuditSecurityEventType</i> defined in 6.4.7, which means it inherits the <i>InstanceDeclarations</i> of that Node.					
HasSubtype	ObjectType	AuditCertificateDataMismatchEventType	Defined in 6.4.13		
HasSubtype	ObjectType	AuditCertificateExpiredEventType	Defined in 6.4.14		
HasSubtype	ObjectType	AuditCertificateInvalidEventType	Defined in 6.4.15		
HasSubtype	ObjectType	AuditCertificateUntrustedEventType	Defined in 6.4.16		
HasSubtype	ObjectType	AuditCertificateRevokedEventType	Defined in 6.4.17		
HasSubtype	ObjectType	AuditCertificateMismatchEventType	Defined in 6.4.18		
HasProperty	Variable	Certificate	ByteString	PropertyType	Mandatory

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

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

6.4.13 AuditCertificateDataMismatchEventType

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

Table 33 – AuditCertificateDataMismatchEventType Definition

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

This *EventType* inherits all *Properties* of the *AuditCertificateEventType*. Their semantic is defined in 6.4.12. The *SourceName* for *Events* of this type should be “Security/Certificate”.

InvalidHostname is the string that represents the host name passed in as part of the URL that is found to be invalid. If the host name was not invalid it can be null.

InvalidUri is the URI that was passed in and found to not match what is contained in the certificate. If the URI was not invalid it can be null.

Either the *InvalidHostname* or *InvalidUri* shall be provided.

6.4.14 AuditCertificateExpiredEventType

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

Table 34 – AuditCertificateExpiredEventType Definition

Attribute		Value			
BrowseName		AuditCertificateExpiredEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditCertificateEventType</i> defined in 6.4.12, which means it inherits the InstanceDeclarations of that Node.					

This *EventType* inherits all *Properties* of the *AuditCertificateEventType*. Their semantic is defined in 6.4.12. The *SourceName* for *Events* of this type should be “Security/Certificate”. The *Message Variable* shall include a description of why the certificate was expired (i.e. time before start or time after end). There are no additional *Properties* defined for this *EventType*.

6.4.15 AuditCertificateInvalidEventType

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

Table 35 – AuditCertificateInvalidEventType Definition

Attribute		Value			
BrowseName		AuditCertificateInvalidEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditCertificateEventType</i> defined in 6.4.12, which means it inherits the InstanceDeclarations of that Node.					

This *EventType* inherits all *Properties* of the *AuditCertificateEventType*. Their semantic is defined in 6.4.12. The *SourceName* for *Events* of this type should be “Security/Certificate”. The *Message* shall include a description of why the certificate is invalid. There are no additional *Properties* defined for this *EventType*.

6.4.16 AuditCertificateUntrustedEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 36.

Table 36 – AuditCertificateUntrustedEventType Definition

Attribute		Value			
BrowseName		AuditCertificateUntrustedEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditCertificateEventType</i> defined in 6.4.12, which means it inherits the InstanceDeclarations of that Node.					

This *EventType* inherits all *Properties* of the *AuditCertificateEventType*. Their semantic is defined in 6.4.12. The *SourceName* for *Events* of this type should be “Security/Certificate”. The *Message Variable* shall include a description of why the certificate is not trusted. If a trust chain is involved then the certificate that failed in the trust chain should be described. There are no additional *Properties* defined for this *EventType*.

6.4.17 AuditCertificateRevokedEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 37.

Table 37 – AuditCertificateRevokedEventType Definition

Attribute		Value			
BrowseName		AuditCertificateRevokedEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditCertificateEventType</i> defined in 6.4.12, which means it inherits the InstanceDeclarations of that Node.					

This *EventType* inherits all *Properties* of the *AuditCertificateEventType*. Their semantic is defined in 6.4.12. The *SourceName* for *Events* of this type should be “Security/Certificate”. The *Message Variable* shall include a description of why the certificate is revoked (was the revocation list unavailable or was the certificate on the list). There are no additional *Properties* defined for this *EventType*.

6.4.18 AuditCertificateMismatchEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 38.

Table 38 – AuditCertificateMismatchEventType Definition

Attribute		Value			
BrowseName		AuditCertificateMismatchEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditCertificateEventType</i> defined in 6.4.12, which means it inherits the InstanceDeclarations of that Node.					

This *EventType* inherits all *Properties* of the *AuditCertificateEventType*. Their semantic is defined in 6.4.12. The *SourceName* for *Events* of this type should be “Security/Certificate”. The *Message Variable* shall include a description of misuse of the certificate. There are no additional *Properties* defined for this *EventType*.

6.4.19 AuditNodeManagementEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 39.

Table 39 – AuditNodeManagementEventType Definition

Attribute		Value			
BrowseName		AuditNodeManagementEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditEventType</i> defined in 6.4.3, which means it inherits the InstanceDeclarations of that Node.					
HasSubtype	ObjectType	AuditAddNodesEventType			
HasSubtype	ObjectType	AuditDeleteNodesEventType			
HasSubtype	ObjectType	AuditAddReferencesEventType			
HasSubtype	ObjectType	AuditDeleteReferencesEventType			

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 *SourceNode* for *Events* of this type should be assigned to the *Server Object*. The *SourceName* for *Events* of this type should be “NodeManagement” and the *Service* that generates the *Event* (e.g. *AddNodes*, *AddReferences*, *DeleteNodes*, *DeleteReferences*).

6.4.20 AuditAddNodesEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 40.

Table 40 – AuditAddNodesEventType Definition

Attribute		Value			
BrowseName		AuditAddNodesEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditNodeManagementEventType</i> defined in 6.4.19, which means it inherits the InstanceDeclarations of that Node.					
HasProperty	Variable	NodesToAdd	AddNodesItem[]	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *AuditNodeManagementEventType*. Their semantic is defined in 6.4.19. The *SourceName* for *Events* of this type should be “NodeManagement/AddNodes”.

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

NodesToAdd is the *NodesToAdd* parameter of the *AddNodes* *Service* call.

6.4.21 AuditDeleteNodesEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 41.

Table 41 – AuditDeleteNodesEventType Definition

Attribute		Value			
BrowseName		AuditDeleteNodesEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditNodeManagementEventType</i> defined in 6.4.19, i.e. inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	NodesToDelete	DeleteNodesItem[]	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *AuditNodeManagementEventType*. Their semantic is defined in 6.4.19. The *SourceName* for *Events* of this type should be “NodeManagement/DeleteNodes”.

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

NodesToDelete is the *nodesToDelete* parameter of the *DeleteNodes* *Service* call.

6.4.22 AuditAddReferencesEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 42.

Table 42 – AuditAddReferencesEventType Definition

Attribute		Value			
BrowseName		AuditAddReferencesEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditNodeManagementEventType</i> defined in 6.4.19, which means it inherits the InstanceDeclarations of that Node.					
HasProperty	Variable	ReferencesToAdd	AddReferencesItem[]	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *AuditNodeManagementEventType*. Their semantic is defined in 6.4.19. The *SourceName* for *Events* of this type should be “NodeManagement/AddReferences”.

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

ReferencesToAdd is the referencesToAdd parameter of the AddReferences *Service* call.

6.4.23 AuditDeleteReferencesEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 43.

Table 43 – AuditDeleteReferencesEventType Definition

Attribute		Value			
BrowseName		AuditDeleteReferencesEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditNodeManagementEventType</i> defined in 6.4.19, which means it inherits the InstanceDeclarations of that Node.					
HasProperty	Variable	ReferencesToDelete	DeleteReferencesItem[]	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *AuditNodeManagementEventType*. Their semantic is defined in 6.4.19. The *SourceName* for *Events* of this type should be “NodeManagement/DeleteReferences”.

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

ReferencesToDelete is the referencesToDelete parameter of the DeleteReferences *Service* call.

6.4.24 AuditUpdateEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 44.

Table 44 – AuditUpdateEventType Definition

Attribute		Value			
BrowseName		AuditUpdateEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditEventType</i> defined in 6.4.3, which means it inherits the InstanceDeclarations of that Node.					
HasSubtype	ObjectType	AuditWriteUpdateEventType	Defined in 6.4.25		
HasSubtype	ObjectType	AuditHistoryUpdateEventType	Defined in 6.4.26		

This *EventType* inherits all *Properties* of the *AuditEventType*. Their semantic is defined in 6.4.3. The *SourceNode* for *Events* of this type should be assigned to the *NodeId* that was changed. The *SourceName* for *Events* of this type should be “Attribute/” and the *Service* that generated the event (e.g. *Write*, *HistoryUpdate*). Note that one *Service* call may generate several *Events* of this type, one per changed value.

6.4.25 AuditWriteUpdateEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 45.

Table 45 – AuditWriteUpdateEventType Definition

Attribute		Value			
BrowseName		AuditWriteUpdateEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditUpdateEventType</i> defined in 6.4.24, which means it inherits the InstanceDeclarations of that Node.					
HasProperty	Variable	Attributeld	UInt32	PropertyType	Mandatory
HasProperty	Variable	IndexRange	NumericRange	PropertyType	Mandatory
HasProperty	Variable	NewValue	BaseDataType	PropertyType	Mandatory
HasProperty	Variable	OldValue	BaseDataType	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *AuditUpdateEventType*. The *SourceName* for *Events* of this type should be “Attribute/Write”. Their semantic is defined in 6.4.24.

Attributeld identifies the *Attribute* that was written on the *SourceNode*.

IndexRange identifies the index range of the written *Attribute* if the *Attribute* is an array. If the *Attribute* is not an array or the whole array was written, the *IndexRange* is set to null.

NewValue identifies the value that was written to the *SourceNode*. If the *IndexRange* is provided, only the values in the provided range are shown.

OldValue identifies the value that the *SourceNode* contained before the write. If the *IndexRange* is provided, only the value of that range is shown. It is acceptable for a *Server* that does not have this information to report a null value.

Both the *NewValue* and the *OldValue* will contain a value in the *DataType* and encoding used for writing the value.

6.4.26 AuditHistoryUpdateEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 46.

Table 46 – AuditHistoryUpdateEventType Definition

Attribute		Value			
BrowseName		AuditHistoryUpdateEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditUpdateEventType</i> defined in 6.4.24, which means it inherits the InstanceDeclarations of that Node.					
HasProperty	Variable	ParameterDataTypeld	NodeId	PropertyType	New

This *EventType* inherits all *Properties* of the *AuditUpdateEventType*. Their semantic is defined in 6.4.24.

The *ParameterDataTypeId* identifies the *DataTypeId* for the extensible parameter used by the *HistoryUpdate*. This parameter indicates the type of *HistoryUpdate* being performed.

Subtypes of this *EventType* are defined in IEC 62541-11 representing the different possibilities to manipulate historical data.

6.4.27 AuditUpdateMethodEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 47.

Table 47 – AuditUpdateMethodEventType Definition

Attribute	Value				
BrowseName	AuditUpdateMethodEventType				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditEventType</i> defined in 6.4.3, which means it inherits the <i>InstanceDeclarations</i> of that Node.					
HasProperty	Variable	MethodId	NodeId	PropertyType	Mandatory
HasProperty	Variable	InputArguments	BaseDataType[]	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *AuditEventType*. Their semantic is defined in 6.4.3. The *SourceNode* for *Events* of this type should be assigned to the *NodeId* of the object that the method resides on. The *SourceName* for *Events* of this type should be "Attribute/Call". Note that one *Service* call may generate several *Events* of this type, one per method called. This *EventType* should be further subtyped to better reflect the functionality of the method and to reflect changes to the address space or updated values triggered by the method.

MethodId identifies the method that was called.

InputArguments identifies the input Arguments for the method. This parameter can be null if no input arguments were provided.

6.4.28 SystemEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 48.

Table 48 – SystemEventType Definition

Attribute	Value				
BrowseName	SystemEventType				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasSubtype	ObjectType	DeviceFailureEventType	Defined in 6.4.29		
HasSubtype	ObjectType	SystemStatusChangeEvent	Defined in 6.4.30		
Subtype of the <i>BaseEventType</i> defined in 6.4.2, which means it inherits the InstanceDeclarations of that Node.					

This *EventType* inherits all *Properties* of the *BaseEventType*. Their semantic is defined in 6.4.2. There are no additional *Properties* defined for this *EventType*.

6.4.29 DeviceFailureEventType

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 49.

Table 49 – DeviceFailureEventType Definition

Attribute		Value			
BrowseName		DeviceFailureEventType			
IsAbstract		True			
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
Subtype of the <i>SystemEventType</i> defined in 6.4.28, which means it inherits the InstanceDeclarations of that Node.					

This *EventType* inherits all *Properties* of the *SystemEventType*. Their semantic is defined in 6.4.28. There are no additional *Properties* defined for this *EventType*.

6.4.30 SystemStatusChangeEvent

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 50.

Table 50 – SystemStatusChangeEvent Definition

Attribute		Value			
BrowseName		SystemStatusChangeEvent			
IsAbstract		True			
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
Subtype of the <i>SystemEventType</i> defined in 6.4.28, which means it inherits the InstanceDeclarations of that Node.					
HasProperty	Variable	SystemState	ServerState	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *SystemEventType*. Their semantic is defined in 6.4.28. The *SourceNode* and the *SourceName* shall identify the system. The system can be the *Server* itself or some underlying system.

The *SystemState* specifies the current state of the system. Changes to the *ServerState* of the system shall trigger a *SystemStatusChangeEvent*, when the event is supported by the system.

6.4.31 BaseModelChangeEvent

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 51.

Table 51 – BaseModelChangeEvent Definition

Attribute	Value				
BrowseName	BaseModelChangeEvent				
IsAbstract	True				
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
Subtype of the <i>BaseEventType</i> defined in 6.4.2, which means it inherits the InstanceDeclarations of that Node.					
HasSubtype	ObjectType	GeneralModelChangeEvent	Defined in 6.4.32		

This *EventType* inherits all *Properties* of the *BaseEventType*. Their semantic is defined in 6.4.2. There are no additional *Properties* defined for this *EventType*. The *SourceNode* for Events of this type should be the *Node* of the *View* that gives the context of the changes. If the whole *AddressSpace* is the context, the *SourceNode* is set to the *NodeId* of the *Server Object*. The *SourceName* for Events of this type should be the *String* part of the *BrowseName* of the *View*; for the whole *AddressSpace* it should be “Server”.

6.4.32 GeneralModelChangeEvent

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 52.

Table 52 – GeneralModelChangeEventDefinition

Attribute	Value				
BrowseName	GeneralModelChangeEvent				
IsAbstract	True				
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
Subtype of the <i>BaseModelChangeEvent</i> defined in 6.4.31, which means it inherits the InstanceDeclarations of that Node.					
HasProperty	Variable	Changes	ModelChangeStructureDataType[]	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *BaseModelChangeEvent*. Their semantic is defined in 6.4.31.

The additional *Property* defined for this *EventType* reflects the changes that issued the *ModelChangeEvent*. It shall contain at least one entry in its array. Its structure is defined in 12.16.

6.4.33 SemanticChangeEventDefinition

This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined in Table 53.

Table 53 – SemanticChangeEventDefinition

Attribute	Value				
BrowseName	SemanticChangeEvent				
IsAbstract	True				
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
Subtype of the <i>BaseEventType</i> defined in 6.4.2, which means it inherits the InstanceDeclarations of that Node.					
HasProperty	Variable	Changes	SemanticChangeStructureDataType[]	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *BaseEventType*. Their semantic is defined in 6.4.2. There are no additional *Properties* defined for this *EventType*. The *SourceNode* for Events of this type should be the *Node* of the *View* that gives the context of the changes. If the whole *AddressSpace* is the context, the *SourceNode* is set to the *NodeId* of the *Server Object*. The *SourceName* for Events of this type should be the *String* part of the *BrowseName* of the *View*, for the whole *AddressSpace* it should be “Server”.

The additional *Property* defined for this *EventType* reflects the changes that issued the *SemanticChangeEvent*. Its structure is defined in 12.17.

6.4.34 EventQueueOverflowEventType

EventQueueOverflow Events are generated when an internal queue of a *MonitoredItem* subscribing for *Events* in the *Server* overflows. IEC 62541-4 defines when the internal *EventQueueOverflow Events* shall be generated.

The *EventType* for *EventQueueOverflow Events* is formally defined in Table 54.

Table 54 – EventQueueOverflowEventType Definition

Attribute	Value				
BrowseName	EventQueueOverflowEvent				
IsAbstract	True				
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
Subtype of the <i>BaseEventType</i> defined in 6.4.2, which means it inherits the InstanceDeclarations of that Node.					

This *EventType* inherits all *Properties* of the *BaseEventType*. Their semantic is defined in 6.4.2. The *SourceNode* for *Events* of this type shall be assigned to the *NodeId* of the *Server Object*. The *SourceName* for *Events* of this type shall be “Internal/EventQueueOverflow”.

6.4.35 ProgressEventType

ProgressEvents are generated to identify the progress of an operation. An operation can be a *Service* call or something application specific like a program execution.

The *EventType* for *Progress Events* is formally defined in Table 55.

Table 55 – ProgressEventType Definition

Attribute	Value				
BrowseName	ProgressEventType				
IsAbstract	True				
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
Subtype of the <i>BaseEventType</i> defined in 6.4.2, which means it inherits the InstanceDeclarations of that Node.					
HasProperty	Variable	Context	BaseDataType	PropertyType	Mandatory
HasProperty	Variable	Progress	UInt16	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *BaseEventType*. Their semantic is defined in 6.4.2. The *SourceNode* for *Events* of this type shall be assigned to the *NodeId* of the *Session Object* where the operation was initiated. The *SourceName* for *Events* of this type shall be “Service/<Service Name as defined in IEC 62541-4>” when the progress of a *Service* call is exposed.

The additional *Property Context* contains context information about what operation progress is reported. In the case of *Service* calls it shall be a *UInt32* containing the *requestHandle* of the *RequestHeader* of the *Service* call.

The additional *Property Progress* contains the percentage completed of the progress. The value shall be between 0 and 100, where 100 identifies that the operation has been finished.

It is recommended that *Servers* only expose *ProgressEvents* for *Service* calls to the *Session* that invoked the *Service*.

6.5 ModellingRuleType

ModellingRules are defined in IEC 62541-3. This *ObjectType* is used as the type for the *ModellingRules*. It is formally defined in Table 56.

Table 56 – ModellingRuleType Definition

Attribute	Value				
BrowseName	ModellingRuleType				
IsAbstract	False				
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
Subtype of the <i>BaseObjectType</i> defined in 6.2					
HasProperty	Variable	NamingRule	NamingRuleType	PropertyType	Mandatory

The *Property NamingRule* identifies the *NamingRule* of a *ModellingRule* as defined in IEC 62541-3.

6.6 FolderType

Instances of this *ObjectType* are used to organise the *AddressSpace* into a hierarchy of *Nodes*. They represent the *root Node* of a subtree, and have no other semantics associated