

INTERNATIONAL STANDARD

NORME INTERNATIONALE



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

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

Withdrawn



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2011 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 la CEI ou du Comité national de la CEI du pays du demandeur.

Si vous avez des questions sur le copyright de la CEI ou si vous désirez obtenir des droits supplémentaires sur cette publication, utilisez les coordonnées ci-après ou contactez le Comité national de la CEI de votre pays de résidence.

IEC Central Office
3, rue de Varembe
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: 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.

- Catalogue of IEC publications: www.iec.ch/searchpub

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 It also gives information on projects, withdrawn and replaced publications.

- IEC Just Published: www.iec.ch/online_news/justpub

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

- Electropedia: www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.

- Customer Service Centre: www.iec.ch/webstore/custserv

If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:

Email: csc@iec.ch

Tel.: +41 22 919 02 11

Fax: +41 22 919 03 00

A propos de la CEI

La Commission Electrotechnique Internationale (CEI) 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 CEI

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

- Catalogue des publications de la CEI: www.iec.ch/searchpub/cur_fut-f.htm

Le Catalogue en-ligne de la CEI vous permet d'effectuer des recherches en utilisant différents critères (numéro de référence, texte, comité d'études,...). Il donne aussi des informations sur les projets et les publications retirées ou remplacées.

- Just Published CEI: www.iec.ch/online_news/justpub

Restez informé sur les nouvelles publications de la CEI. Just Published détaille deux fois par mois les nouvelles publications parues. Disponible en-ligne et aussi par email.

- Electropedia: www.electropedia.org

Le premier dictionnaire en ligne au monde de termes électroniques et électriques. Il contient plus de 20 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans les langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International en ligne.

- Service Clients: www.iec.ch/webstore/custserv/custserv_entry-f.htm

Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions, visitez le FAQ du Service clients ou contactez-nous:

Email: csc@iec.ch

Tél.: +41 22 919 02 11

Fax: +41 22 919 03 00

INTERNATIONAL STANDARD

NORME INTERNATIONALE



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

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

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

COMMISSION
ELECTROTECHNIQUE
INTERNATIONALE

PRICE CODE
CODE PRIX

XE

ICS 25.040.40; 25.100.01

ISBN 978-2-88912-729-0

CONTENTS

FOREWORD.....	10
INTRODUCTION.....	12
1 Scope.....	13
2 Normative references	13
3 Terms, definitions, abbreviations and conventions.....	13
3.1 Terms and definitions	13
3.2 Abbreviations	13
3.3 Conventions for Node descriptions	13
4 NodeIds and BrowseNames.....	15
4.1 NodeIds	15
4.2 BrowseNames	15
5 Common Attributes	16
5.1 General.....	16
5.2 Objects.....	16
5.3 Variables.....	16
5.4 VariableTypes	16
6 Standard ObjectTypes	17
6.1 General.....	17
6.2 BaseObjectType.....	17
6.3 ObjectTypes for the server object.....	17
6.3.1 ServerType.....	17
6.3.2 ServerCapabilitiesType.....	19
6.3.3 ServerDiagnosticsType.....	20
6.3.4 SessionsDiagnosticsSummaryType	21
6.3.5 SessionDiagnosticsObjectType.....	22
6.3.6 VendorServerInfoType.....	22
6.3.7 ServerRedundancyType	22
6.3.8 TransparentRedundancyType	23
6.3.9 NonTransparentRedundancyType.....	23
6.4 ObjectTypes used as EventTypes.....	24
6.4.1 General	24
6.4.2 BaseEventType	24
6.4.3 AuditEventType	26
6.4.4 AuditSecurityEventType	27
6.4.5 AuditChannelEventType	27
6.4.6 AuditOpenSecureChannelEventType	28
6.4.7 AuditSessionEventType.....	28
6.4.8 AuditCreateSessionEventType	29
6.4.9 AuditUrlMismatchEventType	30
6.4.10 AuditActivateSessionEventType	30
6.4.11 AuditCancelEventType	30
6.4.12 AuditCertificateEventType	31
6.4.13 AuditCertificateDataMismatchEventType	31
6.4.14 AuditCertificateExpiredEventType.....	31

6.4.15	AuditCertificateInvalidEventType	32
6.4.16	AuditCertificateUntrustedEventType	32
6.4.17	AuditCertificateRevokedEventType	33
6.4.18	AuditCertificateMismatchEventType	33
6.4.19	AuditNodeManagementEventType	33
6.4.20	AuditAddNodesEventType	34
6.4.21	AuditDeleteNodesEventType	34
6.4.22	AuditAddReferencesEventType	34
6.4.23	AuditDeleteReferencesEventType	35
6.4.24	AuditUpdateEventType	35
6.4.25	AuditWriteUpdateEventType	36
6.4.26	AuditHistoryUpdateEventType	36
6.4.27	AuditUpdateMethodEventType	37
6.4.28	SystemEventType	37
6.4.29	DeviceFailureEventType	37
6.4.30	BaseModelChangeEvent	38
6.4.31	GeneralModelChangeEvent	38
6.4.32	SemanticChangeEvent	38
6.4.33	EventQueueOverflowEventType	39
6.5	ModellingRuleType	39
6.6	FolderType	39
6.7	DataTypeEncodingType	40
6.8	DataTypeSystemType	40
6.9	AggregateFunctionType	40
7	Standard VariableTypes	41
7.1	General	41
7.2	BaseVariableType	41
7.3	PropertyType	41
7.4	BaseDataVariableType	41
7.5	ServerVendorCapabilityType	42
7.6	DataTypeDictionaryType	42
7.7	DataTypeDescriptionType	43
7.8	ServerStatusType	43
7.9	BuildInfoType	43
7.10	ServerDiagnosticsSummaryType	44
7.11	SamplingIntervalDiagnosticsArrayType	44
7.12	SamplingIntervalDiagnosticsType	45
7.13	SubscriptionDiagnosticsArrayType	45
7.14	SubscriptionDiagnosticsType	45
7.15	SessionDiagnosticsArrayType	46
7.16	SessionDiagnosticsVariableType	47
7.17	SessionSecurityDiagnosticsArrayType	49
7.18	SessionSecurityDiagnosticsType	49
8	Standard Objects and their Variables	50
8.1	General	50
8.2	Objects used to organise the AddressSpace structure	50
8.2.1	Overview	50
8.2.2	Root	51
8.2.3	Views	51

8.2.4	Objects	52
8.2.5	Types	52
8.2.6	ObjectTypes	53
8.2.7	VariableTypes	53
8.2.8	ReferenceTypes	54
8.2.9	DataTypes	55
8.2.10	OPC Binary	56
8.2.11	XML Schema	57
8.2.12	EventTypes	57
8.3	Server Object and its containing Objects	58
8.3.1	General	58
8.3.2	Server Object	59
8.4	ModellingRule Objects	60
8.4.1	ExposesItsArray	60
8.4.2	Mandatory	60
8.4.3	Optional	60
9	Standard Methods	61
10	Standard Views	61
11	Standard ReferenceTypes	61
11.1	References	61
11.2	HierarchicalReferences	61
11.3	NonHierarchicalReferences	61
11.4	HasChild	62
11.5	Aggregates	62
11.6	Organizes	62
11.7	HasComponent	63
11.8	HasOrderedComponent	63
11.9	HasProperty	63
11.10	HasSubtype	63
11.11	HasModellingRule	64
11.12	HasTypeDefinition	64
11.13	HasEncoding	64
11.14	HasDescription	64
11.15	HasEventSource	65
11.16	HasNotifier	65
11.17	GeneratesEvent	65
11.18	AlwaysGeneratesEvent	65
11.19	HasModelParent	66
12	Standard DataTypes	66
12.1	Overview	66
12.2	DataTypes defined in Part 3	66
12.3	DataTypes defined in Part 4	70
12.4	BuildInfo	71
12.5	RedundancySupport	72
12.6	ServerState	72
12.7	RedundantServerDataType	73
12.8	SamplingIntervalDiagnosticsDataType	73
12.9	ServerDiagnosticsSummaryDataType	74

12.10 ServerStatusDataType	75
12.11 SessionDiagnosticsDataType	75
12.12 SessionSecurityDiagnosticsDataType.....	76
12.13 ServiceCounterDataType.....	77
12.14 StatusResult.....	77
12.15 SubscriptionDiagnosticsDataType	78
12.16 ModelChangeStructureDataType	79
12.17 SemanticChangeStructureDataType	80
Annex A (informative) Design decisions when modelling the server information	81
Annex B (normative) StateMachines	84
Bibliography.....	103
Figure 1 – Standard AddressSpace Structure	50
Figure 2 – Views Organization	51
Figure 3 – Objects Organization.....	52
Figure 4 – ObjectTypes Organization	53
Figure 5 – VariableTypes Organization	54
Figure 6 – ReferenceType Definitions	55
Figure 7 – DataTypes Organization.....	56
Figure 8 – EventTypes Organization	57
Figure 9 – Excerpt of Diagnostic Information of the Server.....	59
Figure B.1 – Example of a simple state machine.....	85
Figure B.2 – Example of a state machine having a sub-machine	85
Figure B.3 – The StateMachine Information Model.....	87
Figure B.4 – Example of an initial State in a sub-machine.....	92
Figure B.5 – Example of a StateMachineType using inheritance	98
Figure B.6 – Example of a StateMachineType with a SubStateMachine using inheritance.....	99
Figure B.7 – Example of a StateMachineType using containment.....	100
Figure B.8 – Example of a state machine with transitions from sub-states.....	101
Figure B.9 – Example of a StateMachineType having Transitions to SubStateMachines	102
Table 1 – Examples of DataTypes.....	14
Table 2 – Type Definition Table	15
Table 3 – Common Node Attributes	16
Table 4 – Common Object Attributes.....	16
Table 5 – Common Variable Attributes.....	16
Table 6 – Common VariableType Attributes	17
Table 7 – BaseObjectType Definition	17
Table 8 – ServerType Definition.....	18
Table 9 – ServerCapabilitiesType Definition.....	19
Table 10 – ServerDiagnosticsType Definition.....	20
Table 11 – SessionsDiagnosticsSummaryType Definition	21
Table 12 – SessionDiagnosticsObjectType Definition.....	22

Table 13 – VendorServerInfoType Definition	22
Table 14 – ServerRedundancyType Definition.....	23
Table 15 – TransparentRedundancyType Definition	23
Table 16 – NonTransparentRedundancyType Definition.....	23
Table 17 – BaseEventType Definition	24
Table 18 – AuditEventType Definition	26
Table 19 – AuditSecurityEventType Definition.....	27
Table 20 – AuditChannelEventType Definition	27
Table 21 – AuditOpenSecureChannelEventType Definition	28
Table 22 – AuditSessionEventType Definition.....	29
Table 23 – AuditCreateSessionEventType Definition.....	29
Table 24 – AuditUrlMismatchEventType Definition	30
Table 25 – AuditActivateSessionEventType Definition.....	30
Table 26 – AuditCancelEventType Definition	31
Table 27 – AuditCertificateEventType Definition	31
Table 28 – AuditCertificateDataMismatchEventType Definition.....	31
Table 29 – AuditCertificateExpiredEventType Definition.....	32
Table 30 – AuditCertificateInvalidEventType Definition	32
Table 31 – AuditCertificateUntrustedEventType Definition	32
Table 32 – AuditCertificateRevokedEventType Definition.....	33
Table 33 – AuditCertificateMismatchEventType Definition.....	33
Table 34 – AuditNodeManagementEventType Definition	33
Table 35 – AuditAddNodesEventType Definition.....	34
Table 36 – AuditDeleteNodesEventType Definition	34
Table 37 – AuditAddReferencesEventType Definition.....	35
Table 38 – AuditDeleteReferencesEventType Definition.....	35
Table 39 – AuditUpdateEventType Definition	35
Table 40 – AuditWriteUpdateEventType Definition	36
Table 41 – AuditHistoryUpdateEventType Definition	36
Table 42 – AuditUpdateMethodEventType Definition.....	37
Table 43 – SystemEventType Definition.....	37
Table 44 – DeviceFailureEventType Definition	38
Table 45 – BaseModelChangeEventDefinition	38
Table 46 – GeneralModelChangeEventDefinition.....	38
Table 47 – SemanticChangeEventDefinition	39
Table 48 – EventQueueEventType Definition	39
Table 49 – ModellingRuleType Definition	39
Table 50 – FolderType Definition	40
Table 51 – DataTypeEncodingType Definition.....	40
Table 52 – DataTypeSystemType Definition.....	40
Table 53 – AggregateFunctionType Definition.....	40
Table 54 – BaseVariableType Definition	41
Table 55 – PropertyType Definition	41

Table 56 – BaseDataVariableType Definition	42
Table 57 – ServerVendorCapabilityType Definition	42
Table 58 – DataTypeDictionaryType Definition.....	42
Table 59 – DataTypeDescriptionType Definition.....	43
Table 60 – ServerStatusType Definition	43
Table 61 – BuildInfoType Definition	44
Table 62 – ServerDiagnosticsSummaryType Definition	44
Table 63 – SamplingIntervalDiagnosticsArrayType Definition	45
Table 64 – SamplingIntervalDiagnosticsType Definition	45
Table 65 – SubscriptionDiagnosticsArrayType Definition.....	45
Table 66 – SubscriptionDiagnosticsType Definition	46
Table 67 – SessionDiagnosticsArrayType Definition	46
Table 68 – SessionDiagnosticsVariableType Definition	48
Table 69 – SessionSecurityDiagnosticsArrayType Definition.....	49
Table 70 – SessionSecurityDiagnosticsType Definition	50
Table 71 – Root Definition	51
Table 72 – Views Definition	51
Table 73 – Objects Definition	52
Table 74 – Types Definition	52
Table 75 – ObjectTypes Definition	53
Table 76 – VariableTypes Definition.....	54
Table 77 – ReferenceTypes Definition	55
Table 78 – DataTypes Definition	56
Table 79 – OPC Binary Definition.....	57
Table 80 – XML Schema Definition	57
Table 81 – EventTypes Definition	58
Table 82 – Server Definition	60
Table 83 – ExposesItsArray Definition	60
Table 84 – Mandatory Definition	60
Table 85 – Optional Definition.....	60
Table 86 – References ReferenceType	61
Table 87 – HierarchicalReferences ReferenceType.....	61
Table 88 – NonHierarchicalReferences ReferenceType	62
Table 89 – HasChild ReferenceType.....	62
Table 90 – Aggregates ReferenceType	62
Table 91 – Organizes ReferenceType	62
Table 92 – HasComponent ReferenceType	63
Table 93 – HasOrderedComponent ReferenceType	63
Table 94 – HasProperty ReferenceType.....	63
Table 95 – HasSubtype ReferenceType	63
Table 96 – HasModellingRule ReferenceType.....	64
Table 97 – HasTypeDefinition ReferenceType	64
Table 98 – HasEncoding ReferenceType	64

Table 99 – HasDescription ReferenceType	64
Table 100 – HasEventSource ReferenceType	65
Table 101 – HasNotifier ReferenceType	65
Table 102 – GeneratesEvent ReferenceType	65
Table 103 – AlwaysGeneratesEvent ReferenceType	65
Table 104 – HasModelParent ReferenceType	66
Table 105 – Part 3 DataType Definitions	67
Table 106 – BaseDataType Definition	68
Table 107 – Structure Definition	68
Table 108 – Enumeration Definition	69
Table 109 – ByteString Definition	69
Table 110 – Number Definition	69
Table 111 – Double Definition	69
Table 112 – Integer Definition	69
Table 113 – DateTime Definition	70
Table 114 – String Definition	70
Table 115 – UInteger Definition	70
Table 116 – Image Definition	70
Table 117 – Part 4 DataType Definitions	71
Table 118 – UserIdentityToken Definition	71
Table 119 – BuildInfo Structure	72
Table 120 – BuildInfo Definition	72
Table 121 – RedundancySupport Values	72
Table 122 – RedundancySupport Definition	72
Table 123 – ServerState Values	73
Table 124 – ServerState Definition	73
Table 125 – RedundantServerDataType Structure	73
Table 126 – RedundantServerDataType Definition	73
Table 127 – SamplingIntervalDiagnosticsDataType Structure	74
Table 128 – SamplingIntervalDiagnosticsDataType Definition	74
Table 129 – ServerDiagnosticsSummaryDataType Structure	74
Table 130 – ServerDiagnosticsSummaryDataType Definition	74
Table 131 – ServerStatusDataType Structure	75
Table 132 – ServerStatusDataType Definition	75
Table 133 – SessionDiagnosticsDataType Structure	75
Table 134 – SessionDiagnosticsDataType Definition	76
Table 135 – SessionSecurityDiagnosticsDataType Structure	77
Table 136 – SessionSecurityDiagnosticsDataType Definition	77
Table 137 – ServiceCounterDataType Structure	77
Table 138 – ServiceCounterDataType Definition	77
Table 139 – StatusResult Structure	78
Table 140 – StatusResult Definition	78
Table 141 – SubscriptionDiagnosticsDataType Structure	79

Table 142 – SubscriptionDiagnosticsDataType Definition.....	79
Table 143 – ModelChangeStructureDataType Structure.....	80
Table 144 – ModelChangeStructureDataType Definition	80
Table 145 – SemanticChangeStructureDataType Structure.....	80
Table 146 – SemanticChangeStructureDataType Definition	80
Table B.1 – StateMachineType Definition.....	88
Table B.2 – StateVariableType Definition.....	88
Table B.3 – TransitionVariableType Definition.....	89
Table B.4 – FiniteStateMachineType Definition	90
Table B.5 – FiniteStateVariableType Definition	91
Table B.6 – FiniteTransitionVariableType Definition	91
Table B.7 – StateType Definition.....	92
Table B.8 – InitialStateType Definition	93
Table B.9 – TransitionType Definition	93
Table B.10 – FromState ReferenceType	93
Table B.11 – ToState ReferenceType	94
Table B.12 – HasCause ReferenceType	94
Table B.13 – HasEffect ReferenceType	95
Table B.14 – HasSubStateMachine ReferenceType	95
Table B.15 – TransitionEventType	96
Table B.16 – AuditUpdateStateEventType	96
Table B.17 – Specific StatusCodes for StateMachines	97

Withdrawing

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.

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

FDIS	Report on voting
65E/192/FDIS	65E/214/RVD

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

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

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

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

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

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

Withdrawn

INTRODUCTION

This International Standard is the specification for developers of OPC UA applications. The specification is a result of an analysis and design process to develop a standard interface to facilitate the development of applications by multiple vendors that inter-operate seamlessly together.

Withdrawn

OPC Unified Architecture –

Part 5: Information Model

1 Scope

This part of IEC 62541 defines the Information Model of the OPC Unified Architecture (OPC UA). 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 referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

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

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

3 Terms, definitions, abbreviations and conventions

3.1 Terms and definitions

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

3.1.1

ClientUserId

String that identifies the user of the client requesting an action

NOTE 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

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. That implies that the referenced *Node* has a *HasModelParent* *Reference* with the *Node* defined in the Table as *TargetNode* (see IEC 62541-3 for the definition of *ModelParents*).

- 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 Part 3) and the *ArrayDimensions* is set to null or is omitted. In Table 1 examples are given.

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 Type Definition 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 Part 3.

Note that this standard does not only define 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*. But the *NamespaceIndex* of those *Nodes* cannot be the *NamespaceIndex* 0, because they are not defined by this standard 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 Part 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 Server 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 International 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 Part 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	ModellingRule
HasSubtype	ObjectType	ServerType	Defined in 6.3.1		
HasSubtype	ObjectType	ServerCapabilitiesType	Defined in 6.3.2		
HasSubtype	ObjectType	ServerDiagnosticsType	Defined in 6.3.3		
HasSubtype	ObjectType	SessionsDiagnosticsSummaryType	Defined in 6.3.4		
HasSubtype	ObjectType	SessionDiagnosticsObjectType	Defined in 6.3.5		
HasSubtype	ObjectType	VendorServerInfoType	Defined in 6.3.6		
HasSubtype	ObjectType	ServerRedundancyType	Defined in 6.3.7		
HasSubtype	ObjectType	BaseEventType	Defined in 6.4.2		
HasSubtype	ObjectType	ModellingRuleType	Defined in 6.5		
HasSubtype	ObjectType	FolderType	Defined in 6.6		
HasSubtype	ObjectType	DataTypeEncodingType	Defined in 6.7		
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
¹ 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. Part 4 describes discovery mechanism that can be used to resolve URIs into URLs.

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 Part 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*.

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	Modelling Rule
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	SoftwareCertificate[]	PropertyType	Mandatory
HasComponent	Object	ModellingRules		FolderType	Mandatory
HasComponent	Object	AggregateFunctions		FolderType	Mandatory
HasComponent	Variable	Vendor specific <i>Variables</i> of a subtype of the ServerVendorCapabilityType defined in 7.5			--

ServerProfileArray defines the conformance profile of the server. See Part 7 for the definitions of server profiles.

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 client should

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 ReadHistory 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 ReadHistory 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 SoftwareCertificates containing all certificates supported by the server. This shall be the same list of certificates as returned by the CreateSession Service. The certificates in this Property are not signed.

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 subclass 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	NodeClass	BrowseName	Data Type / TypeDefinition	Modelling-Rule
Subtype of the BaseObjectType defined in 6.2				
HasComponent	Variable	ServerDiagnosticsSummary	ServerDiagnosticsSummaryDataType ServerDiagnosticsSummaryType	Mandatory
HasComponent	Variable	SamplingIntervalDiagnosticsArray	SamplingIntervalDiagnosticsDataType[] SamplingIntervalDiagnosticsArrayType	Optional
HasComponent	Variable	SubscriptionDiagnosticsArray	SubscriptionDiagnosticsDataType[] SubscriptionDiagnosticsArrayType	Mandatory
HasComponent	Object	SessionsDiagnosticsSummary	-- SessionsDiagnosticsSummaryType	Mandatory
HasProperty	Variable	EnabledFlag	Boolean PropertyType	Mandatory

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

SamplingIntervalDiagnosticsArray is an array of diagnostic information per sampling rate as defined in 12.8. There is one entry for each sampling rate currently used by the server. Its *TypeDefinitionNode* is the *VariableType* *SamplingIntervalDiagnosticsArrayType*, providing a *Variable* for each entry in the array, as defined in 7.11.

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	ModellingRule
Subtype of the BaseObjectType defined in 6.2				
HasComponent	Variable	SessionDiagnosticsArray	SessionDiagnosticsDataType[] SessionDiagnosticsArrayType	Mandatory
HasComponent	Variable	SessionSecurityDiagnosticsArray	SessionSecurityDiagnosticsDataType[] SessionSecurityDiagnosticsArrayType	Mandatory
HasComponent	Object	For each session of the server one <i>Object</i> has to be provided ¹	-- SessionDiagnosticsObjectType	--
¹ 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. It has the *ClientName* of the session as *BrowseName* and 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	Data Type	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	TypeDefinition	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.

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	NodeClass	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 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	ModellingRule
Subtype of the ServerRedundancyType defined in 6.3.7, that is, inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	ServerUriArray	String[]	PropertyType	Mandatory

ServerUriArray is an array with the URI of all redundant servers of the OPC UA server. See Part 1 for the definition of redundancy in this standard. Since, in a non-transparent redundancy environment, the client is responsible to subscribe to the redundant servers, it might or might not open a session to one or more redundant servers of this array.

The redundancy support provided by the server is defined in the *RedundancySupport* (defined in the supertype). The client is allowed to access the redundant sever only as described there, however, "hot" switchover implies the support of "warm" switchover and "warm" switchover implies the support of "cold" switchover.

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.

6.4 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 Part 3. The following subclauses specify their representation in the *AddressSpace*.

6.4.2 BaseEventType

This *EventType* is defined in Part 3. Its representation in the *AddressSpace* is formally defined in Table 17.

Table 17 – BaseEventType Definition

Attribute	Value				
BrowseName	BaseEventType				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>BaseObjectType</i> defined in 6.2					
HasSubtype	ObjectType	AuditEventType	Defined in 6.4.3		
HasSubtype	ObjectType	SystemEventType	Defined in 6.4.28		
HasSubtype	ObjectType	BaseModelChangeEventType	Defined in 6.4.30		
HasSubtype	ObjectType	SemanticChangeEventType	Defined in 6.4.32		
HasSubtype	ObjectType	EventQueueOverflowEventType	Defined in 6.4.33		
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.

EventType describes the specific type of *Event*.

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 *DisplayName* of the *Event* source, 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 Part 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*.

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

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

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

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

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

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

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

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

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

The purpose of this approach is to allow clients to use severity values from multiple servers from different vendors in a consistent manner. Additional discussions of severity can be found in Part 9.

6.4.3 AuditEventType

This *EventType* is defined in Part 3. Its representation in the *AddressSpace* is formally defined in Table 18.

Table 18 – AuditEventType Definition

Attribute	Value				
BrowseName	AuditEventType				
IsAbstract	True				
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
Subtype of the <i>BaseEventType</i> defined in 6.4.2, that is, inheriting the <i>InstanceDeclarations</i> 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 Part 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. This token can contain the information in multiple formats depending on the type of User Identity that is passed to the service. If the *UserIdentityToken* that was passed was defined as a *UserName*, then the structure contains an explicit string that is the user. If the passed *UserIdentityToken* was defined as X509v3, then the *CertificateData* byte string contains an element that is the user string which can be extracted from the subject key in this structure. If the passed *UserIdentityToken* was defined as WSS, then the user string can be extracted from the WS-Security XML token. If an *AnonymousIdentityToken* was used, the value is null.

6.4.4 AuditSecurityEventType

This *EventType* is defined in Part 3. Its representation in the *AddressSpace* is formally defined in Table 19.

Table 19 – 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, that is, inheriting the <i>InstanceDeclarations</i> 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 Part 3. Its representation in the *AddressSpace* is formally defined in Table 20.

Table 20 – AuditChannelEventType Definition

Attribute	Value				
BrowseName	AuditChannelEventType				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditSecurityEventType</i> defined in 6.4.4, that is, inheriting 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 (*AuditSessionEventType*).

and its subtypes) and the SecureChannel Service Set (AuditChannelEventType and its subtypes).

6.4.6 AuditOpenSecureChannelEventType

This *EventType* is defined in Part 3. Its representation in the *AddressSpace* is formally defined in Table 21.

Table 21 – AuditOpenSecureChannelEventType Definition

Attribute	Value				
BrowseName	AuditOpenSecureChannelEventType				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the <i>AuditChannelEventType</i> defined in 6.4.5, that is, inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	ClientCertificate	ByteString	PropertyType	Mandatory
HasProperty	Variable	ClientCertificateThumbprint	String	PropertyType	Mandatory
HasProperty	Variable	RequestType	SecurityTokenRequestType	PropertyType	Mandatory
HasProperty	Variable	SecurityPolicyUri	String	PropertyType	Mandatory
HasProperty	Variable	SecurityMode	MessageSecurityMode	PropertyType	Mandatory
HasProperty	Variable	RequestedLifetime	Duration	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *AuditChannelEventType*. Their semantic is defined in 6.4.5. The *SourceName* for *Events* of this type 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*.

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 Part 3. Its representation in the *AddressSpace* is formally defined in Table 22.

Table 22 – 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, that is, inheriting 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. There are no additional *Properties* defined for this *EventType*.

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 Part 3. Its representation in the *AddressSpace* is formally defined in Table 23.

Table 23 – 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, that is, inheriting the InstanceDeclarations of that Node.					
HasSubtype	ObjectType	AuditUnMismatchEventType	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* (*AuditSessionEventType* and its 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*.

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

6.4.9 AuditUrlMismatchEventType

This *EventType* is defined in Part 3. Its representation in the *AddressSpace* is formally defined in Table 23.

Table 24 – 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 that is, inheriting the <i>InstanceDeclarations</i> of that Node.					
HasProperty	Variable	EndpointUrl	String	PropertyType	Mandatory

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

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

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

6.4.10 AuditActivateSessionEventType

This *EventType* is defined in Part 3. Its representation in the *AddressSpace* is formally defined in Table 25.

Table 25 – AuditActivateSessionEventType Definition

Attribute	Value				
BrowseName	AuditActivateSessionEventType				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditSessionEventType</i> defined in 6.4.7, that is, inheriting the <i>InstanceDeclarations</i> of that Node.					
HasProperty	Variable	ClientSoftwareCertificates	SignedSoftwareCertificate[]	PropertyType	Mandatory
HasProperty	Variable	UserIdentityToken	UserIdentityToken	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.

6.4.11 AuditCancelEventType

This *EventType* is defined in Part 3. Its representation in the *AddressSpace* is formally defined in Table 26.

Table 26 – 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 Part 3. Its representation in the *AddressSpace* is formally defined in Table 27.

Table 27 – AuditCertificateEventType Definition

Attribute	Value				
BrowseName	AuditCertificateEventType				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditSecurityEventType</i> defined in 6.4.7, that is, inheriting the InstanceDeclarations 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 Part 3. Its representation in the *AddressSpace* is formally defined in Table 28.

Table 28 – AuditCertificateDataMismatchEventType Definition

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

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 Part 3. Its representation in the *AddressSpace* is formally defined in Table 29.

Table 29 – 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, that is, inheriting 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 Part 3. Its representation in the *AddressSpace* is formally defined in Table 30.

Table 30 – AuditCertificateInvalidEventType Definition

Attribute	Value				
BrowseName	AuditCertificateInvalidEvent type				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>AuditCertificateEventType</i> defined in 6.4.12, that is, inheriting 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 Part 3. Its representation in the *AddressSpace* is formally defined in Table 31.

Table 31 – 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, that is, inheriting 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 Part 3. Its representation in the *AddressSpace* is formally defined in Table 32.

Table 32 – 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, that is, inheriting 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 Part 3. Its representation in the *AddressSpace* is formally defined in Table 33.

Table 33 – 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, that is, inheriting 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 the certificated misuse. There are no additional *Properties* defined for this *EventType*.

6.4.19 AuditNodeManagementEventType

This *EventType* is defined in Part 3. Its representation in the *AddressSpace* is formally defined in Table 34.

Table 34 – 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, that is, inheriting 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 Part 3. Its representation in the *AddressSpace* is formally defined in Table 35.

Table 35 – 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, that is, inheriting the <i>InstanceDeclarations</i> 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 Part 3. Its representation in the *AddressSpace* is formally defined in Table 36.

Table 36 – 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 <i>InstanceDeclarations</i> 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 Part 3. Its representation in the *AddressSpace* is formally defined in Table 37.

Table 37 – 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, that is, inheriting 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 Part 3. Its representation in the *AddressSpace* is formally defined in Table 38.

Table 38 – 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, that is, inheriting 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 Part 3. Its representation in the *AddressSpace* is formally defined in Table 39.

Table 39 – 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, that is, inheriting 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 Part 3. Its representation in the *AddressSpace* is formally defined in Table 40.

Table 40 – 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, that is, inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	AttributeId	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.

AttributeId 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 *AttributeIndexRange* is set to null.

NewValue identifies the value that was written to the *SourceNode*. If the *AttributeIndexRange* is provided, only the value of that range is shown.

OldValue identifies the value that the *SourceNode* contained before the write. If the *AttributeIndexRange* 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 Part 3. Its representation in the *AddressSpace* is formally defined in Table 41.

Table 41 – 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, that is, inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	ParameterDataTypeId	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 Part 11 representing the different possibilities to manipulate historical data.

6.4.27 AuditUpdateMethodEventType

This *EventType* is defined in Part 3. Its representation in the *AddressSpace* is formally defined in Table 42.

Table 42 – 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, that is, inheriting the InstanceDeclarations 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 *AddressSpace* 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 Part 3. Its representation in the *AddressSpace* is formally defined in Table 43.

Table 43 – SystemEventType Definition

Attribute	Value				
BrowseName	SystemEventType				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasSubtype	ObjectType	DeviceFailureEventType	Defined in 6.4.29		
Subtype of the <i>BaseEventType</i> defined in 6.4.2, that is, inheriting 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 Part 3. Its representation in the *AddressSpace* is formally defined in Table 44.

Table 44 – 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, that is, inheriting 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*.

6.4.30 BaseModelChangeEvent Type

This *EventType* is defined in Part 3. Its representation in the *AddressSpace* is formally defined in Table 45.

Table 45 – BaseModelChangeEvent Type Definition

Attribute	Value				
BrowseName	BaseModelChangeEvent				
IsAbstract	True				
References	NodeClass	BrowseName	Data Type	Type Definition	ModellingRule
Subtype of the <i>BaseEventType</i> defined in 6.4.2, that is, inheriting the InstanceDeclarations of that Node.					
HasSubtype	ObjectType	GeneralModelChangeEvent	Defined in 6.4.31		

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”.

6.4.31 GeneralModelChangeEvent Type

This *EventType* is defined in Part 3. Its representation in the *AddressSpace* is formally defined in Table 46.

Table 46 – GeneralModelChangeEvent Type Definition

Attribute	Value				
BrowseName	GeneralModelChangeEvent				
IsAbstract	True				
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
Subtype of the <i>BaseModelChangeEvent</i> defined in 6.4.30, that is, inheriting the InstanceDeclarations of that Node.					
HasProperty	Variable	Changes	ModelChangeStructureDataType[]	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *BaseModelChangeEvent*. Their semantic is defined in 6.4.30.

The additional *Property* defined for this *EventType* reflects the changes that issued the *ModelChangeEvent*. Its structure is defined in 12.16.

6.4.32 SemanticChangeEvent Type

This *EventType* is defined in Part 3. Its representation in the *AddressSpace* is formally defined in Table 47.

Table 47 – SemanticChangeEventType Definition

Attribute	Value				
BrowseName	SemanticChangeEventType				
IsAbstract	True				
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
Subtype of the <i>BaseEventType</i> defined in 6.4.2, that is, inheriting 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.

6.4.33 EventQueueOverflowEventType

EventQueueOverflow Events are generated when an internal queue of a *MonitoredItem* subscribing for *Events* in the server overflows. Part 4 defines when the internal *EventQueueOverflow Events* shall be generated.

The *EventType* for *EventQueueOverflow Events* is formally defined in Table 48.

Table 48 – EventQueueEventType Definition

Attribute	Value				
BrowseName	EventQueueOverflowEventType				
IsAbstract	True				
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
Subtype of the <i>BaseEventType</i> defined in 6.4.2, that is, inheriting 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”.

6.5 ModellingRuleType

ModellingRules are defined in Part 3. This *ObjectType* is used as the type for the *ModellingRules*. It is formally defined in Table 49.

Table 49 – 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 Part 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 with them. However, the *DisplayName* of an instance of the *FolderType* such as

“ObjectTypes”, should imply the semantics associated with the use of it. There are no References specified for this *ObjectType*. It is formally defined in Table 50.

Table 50 – FolderType Definition

Attribute	Value				
BrowseName	FolderType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the BaseObjectType defined in 6.2.					

6.7 DataTypeEncodingType

DataTypeEncodings are defined in Part 3. This *ObjectType* is used as type for the *DataTypeEncodings*. There are no References specified for this *ObjectType*. It is formally defined in Table 51.

Table 51 – DataTypeEncodingType Definition

Attribute	Value				
BrowseName	DataTypeEncodingType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the BaseObjectType defined in 6.2.					

6.8 DataTypeSystemType

DataTypeSystems are defined in Part 3. This *ObjectType* is used as type for the *DataTypeSystems*. There are no References specified for this *ObjectType*. It is formally defined in Table 52.

Table 52 – DataTypeSystemType Definition

Attribute	Value				
BrowseName	DataTypeSystemType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the BaseObjectType defined in 6.2.					

6.9 AggregateFunctionType

This *ObjectType* defines an *AggregateFunction* supported by a UA server. It is formally defined in Table 53.

Table 53 – AggregateFunctionType Definition

Attribute	Value				
BrowseName	AggregateFunctionType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the BaseObjectType defined in 6.2.					

For the *AggregateFunctionType*, the *Description Attribute* is mandatory. The *Description Attribute* provides a localized description of the *AggregateFunction*. Specific *AggregateFunctions* may be defined in further parts of this series of standards.

7 Standard VariableTypes

7.1 General

Typically, the components of a complex *VariableType* are fixed and can be extended by subtyping. However, because each *Variable* of a *VariableType* can be extended with additional components this standard allows the extension of the standard *VariableTypes* defined in this document with additional components. This allows the expression of additional information in the type definition that would be contained in each *Variable* anyway. However, it is not allowed to restrict the components of the standard *VariableTypes* defined in this International Standard. An example of extending *VariableTypes* would be putting the standard *Property NodeVersion*, defined in Part 3, into the *BaseDataVariableType*, stating that each *DataVariable* of the server will provide a *NodeVersion*.

7.2 BaseVariableType

The *BaseVariableType* is the abstract base type for all other *VariableTypes*. However, only the *PropertyType* and the *BaseDataVariableType* directly inherit from this type.

There are no *References*, except for *HasSubtype References*, specified for this *VariableType*. It is formally defined in Table 54.

Table 54 – BaseVariableType Definition

Attribute	Value				
BrowseName	BaseVariableType				
IsAbstract	True				
ValueRank	-2 (-2 = Any)				
DataType	BaseDataType				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasSubtype	VariableType	PropertyType		Defined in 7.3	
HasSubtype	VariableType	BaseDataVariableType		Defined in 7.4	

7.3 PropertyType

The *PropertyType* is a subtype of the *BaseVariableType*. It is used as the type definition for all *Properties*. *Properties* are defined by their *BrowseName* and therefore they do not need a specialised type definition. It is not allowed to subtype this *VariableType*.

There are no *References* specified for this *VariableType*. It is formally defined in Table 55.

Table 55 – PropertyType Definition

Attribute	Value				
BrowseName	PropertyType				
IsAbstract	False				
ValueRank	-2 (-2 = Any)				
DataType	BaseDataType				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the <i>BaseVariableType</i> defined in 7.2.					

7.4 BaseDataVariableType

The *BaseDataVariableType* is a subtype of the *BaseVariableType*. It is used as the type definition whenever there is a *DataVariable* having no more concrete type definition available. This *VariableType* is the base *VariableType* for *VariableTypes* of *DataVariables*, and all other *VariableTypes* of *DataVariables* shall either directly or indirectly inherit from it. However, it might not be possible for servers to provide all *HasSubtype References* from this *VariableType* to its subtypes, and therefore it is not required to provide this information.

There are no *References* except for *HasSubtype References* specified for this *VariableType*. It is formally defined in Table 56.

Table 56 – BaseDataVariableType Definition

Attribute	Value		
BrowseName	BaseDataVariableType		
IsAbstract	False		
ValueRank	-2 (-2 = Any)		
DataType	BaseDataType		
References	NodeClass	BrowseName	Comment
Subtype of the BaseVariableType defined in 7.2.			
HasSubtype	VariableType	ServerVendorCapabilityType	Defined in 7.5
HasSubtype	VariableType	DataTypeDictionaryType	Defined in 7.6
HasSubtype	VariableType	DataTypeDescriptionType	Defined in 7.7
HasSubtype	VariableType	ServerStatusType	Defined in 7.8
HasSubtype	VariableType	BuildInfoType	Defined in 7.9
HasSubtype	VariableType	ServerDiagnosticsSummaryType	Defined in 7.10
HasSubtype	VariableType	SamplingIntervalDiagnosticsArrayType	Defined in 7.11
HasSubtype	VariableType	SamplingIntervalDiagnosticsType	Defined in 7.12
HasSubtype	VariableType	SubscriptionDiagnosticsArrayType	Defined in 7.13
HasSubtype	VariableType	SubscriptionDiagnosticsType	Defined in 7.14
HasSubtype	VariableType	SessionDiagnosticsArrayType	Defined in 7.15
HasSubtype	VariableType	SessionDiagnosticsVariableType	Defined in 7.16
HasSubtype	VariableType	SessionSecurityDiagnosticsArrayType	Defined in 7.17
HasSubtype	VariableType	SessionSecurityDiagnosticsType	Defined in 7.18

7.5 ServerVendorCapabilityType

This *VariableType* is an abstract type whose subtypes define capabilities of the server. Vendors may define subtypes of this type. This *VariableType* is formally defined in Table 57.

Table 57 – ServerVendorCapabilityType Definition

Attribute	Value				
BrowseName	ServerVendorCapabilityType				
IsAbstract	True				
ValueRank	-1 (-1 = Scalar)				
DataType	BaseDataType				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the BaseDataVariableType defined in 7.4.					

7.6 DataTypeDictionaryType

DataTypeDictionaries are defined in Part 3. This *VariableType* is used as the type for the *DataTypeDictionaries*. There are no *References* specified for this *VariableType*. It is formally defined in Table 58.

Table 58 – DataTypeDictionaryType Definition

Attribute	Value				
BrowseName	DataTypeDictionaryType				
IsAbstract	False				
ValueRank	-1 (-1 = Scalar)				
DataType	ByteString				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the BaseDataVariableType defined in 7.4.					
HasProperty	Variable	DataTypeVersion	String	PropertyType	Optional
HasProperty	Variable	NamespaceUri	String	PropertyType	Optional

The meaning of the *Property* *DataTypeVersion* is defined in Part 3. The *NamespaceUri* is the URI for the namespace described by the *Value Attribute* of the *DataTypeDictionary*.

7.7 DataTypeDescriptionType

DataTypeDescriptions are defined in Part 3. This *VariableType* is used as the type for the *DataTypeDescriptions*. There are no *References* specified for this *VariableType*. It is formally defined in Table 59.

Table 59 – DataTypeDescriptionType Definition

Attribute	Value				
BrowseName	DataTypeDescriptionType				
IsAbstract	False				
ValueRank	-1 (-1 = Scalar)				
DataType	ByteString				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the BaseDataVariableType defined in 7.4.					
HasProperty	Variable	DataTypeVersion	String	PropertyType	Optional
HasProperty	Variable	DictionaryFragment	ByteString	PropertyType	Optional

The meaning of the *Properties* *DataTypeVersion* and *DictionaryFragment* is defined in Part 3.

7.8 ServerStatusType

This complex *VariableType* is used for information about the server status. Its *DataVariables* reflect its *DataType* having the same semantic defined in 12.10. The *VariableType* is formally defined in Table 60.

Table 60 – ServerStatusType Definition

Attribute	Value				
BrowseName	ServerStatusType				
IsAbstract	False				
ValueRank	-1 (-1 = Scalar)				
DataType	ServerStatusDataType				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the BaseDataVariableType defined in 7.4.					
HasComponent	Variable	StartTime	UtcTime	BaseDataVariableType	Mandatory
HasComponent	Variable	CurrentTime	UtcTime	BaseDataVariableType	Mandatory
HasComponent	Variable	State	ServerState	BaseDataVariableType	Mandatory
HasComponent	Variable	BuildInfo	BuildInfo	BuildInfoType	Mandatory
HasComponent	Variable	SecondsTillShutdown	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	ShutdownReason	LocalizedText	BaseDataVariableType	Mandatory
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.					

7.9 BuildInfoType

This complex *VariableType* is used for information about the server status. Its *DataVariables* reflect its *DataType* having the same semantic defined in 12.4. The *VariableType* is formally defined in Table 61.

Table 61 – BuildInfoType Definition

Attribute	Value				
BrowseName	BuildInfoType				
IsAbstract	False				
ValueRank	-1 (-1 = Scalar)				
DataType	BuildInfo				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the BaseDataVariableType defined in 7.4.					
HasComponent	Variable	ProductUri	String	BaseDataVariableType	Mandatory
HasComponent	Variable	ManufacturerName	String	BaseDataVariableType	Mandatory
HasComponent	Variable	ProductName	String	BaseDataVariableType	Mandatory
HasComponent	Variable	SoftwareVersion	String	BaseDataVariableType	Mandatory
HasComponent	Variable	BuildNumber	String	BaseDataVariableType	Mandatory
HasComponent	Variable	BuildDate	UtcTime	BaseDataVariableType	Mandatory

7.10 ServerDiagnosticsSummaryType

This complex *VariableType* is used for diagnostic information. Its *DataVariables* reflect its *DataType* having the same semantic defined in 12.9. The *VariableType* is formally defined in Table 62.

Table 62 – ServerDiagnosticsSummaryType Definition

Attribute	Value				
BrowseName	ServerDiagnosticsSummaryType				
IsAbstract	False				
ValueRank	-1 (-1 = Scalar)				
DataType	ServerDiagnosticsSummaryDataType				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the BaseDataVariableType defined in 7.4.					
HasComponent	Variable	ServerViewCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	CurrentSessionCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	CumulatedSessionCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	SecurityRejectedSessionCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	RejectedSessionCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	SessionTimeoutCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	SessionAbortCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	SamplingRateCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	PublishingIntervalCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	CurrentSubscriptionCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	CumulatedSubscriptionCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	SecurityRejectedRequestsCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	RejectedRequestsCount	UInt32	BaseDataVariableType	Mandatory

7.11 SamplingIntervalDiagnosticsArrayType

This complex *VariableType* is used for diagnostic information. For each entry of the array, instances of this type will provide a *Variable* of the *SamplingIntervalDiagnosticsType* *VariableType* having the sampling rate as *BrowseName*. The *VariableType* is formally defined in Table 63.

Table 63 – SamplingIntervalDiagnosticsArrayType Definition

Attribute	Value			
BrowseName	SamplingIntervalDiagnosticsArrayType			
IsAbstract	False			
ValueRank	1 (1 = OneDimension)			
ArrayDimensions	{0} (0 = UnknownSize)			
DataType	SamplingIntervalDiagnosticsDataType			
References	NodeClass	BrowseName	DataType TypeDefinition	ModellingRule
Subtype of the BaseDataVariableType defined in 7.4.				
HasComponent	VariableType	SamplingIntervalDiagnostics	SamplingIntervalDiagnosticsDataType SamplingIntervalDiagnosticsType	ExposesItsArray

7.12 SamplingIntervalDiagnosticsType

This complex *VariableType* is used for diagnostic information. Its *DataVariables* reflect its *DataType*, having the same semantic defined in 12.8. The *VariableType* is formally defined in Table 64.

Table 64 – SamplingIntervalDiagnosticsType Definition

Attribute	Value				
BrowseName	SamplingIntervalDiagnosticsType				
IsAbstract	False				
ValueRank	-1 (-1 = Scalar)				
DataType	SamplingIntervalDiagnosticsDataType				
References	Node Class	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the BaseDataVariableType defined in 7.4.					
HasComponent	Variable	SamplingRate	Duration	BaseDataVariableType	Mandatory
HasComponent	Variable	SampledMonitoredItemsCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	MaxSampledMonitoredItemsCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	DisabledMonitoredItemsSamplingCount	UInt32	BaseDataVariableType	Mandatory

7.13 SubscriptionDiagnosticsArrayType

This complex *VariableType* is used for diagnostic information. For each entry of the array, instances of this type will provide a *Variable* of the SubscriptionDiagnosticsType *VariableType* having the SubscriptionId as *BrowseName*. The *VariableType* is formally defined in Table 65.

Table 65 – SubscriptionDiagnosticsArrayType Definition

Attribute	Value			
BrowseName	SubscriptionDiagnosticsArrayType			
IsAbstract	False			
ValueRank	1 (1 = OneDimension)			
ArrayDimensions	{0} (0 = UnknownSize)			
DataType	SubscriptionDiagnosticsDataType			
References	NodeClass	BrowseName	DataType TypeDefinition	ModellingRule
Subtype of the BaseDataVariableType defined in 7.4.				
HasComponent	VariableType	SubscriptionDiagnostics	SubscriptionDiagnosticsDataType SubscriptionDiagnosticsType	ExposesItsArray

7.14 SubscriptionDiagnosticsType

This complex *VariableType* is used for diagnostic information. Its *DataVariables* reflect its *DataType*, having the same semantic defined in 12.15. The *VariableType* is formally defined in Table 66.

Table 66 – SubscriptionDiagnosticsType Definition

Attribute	Value				
BrowseName	SubscriptionDiagnosticsType				
IsAbstract	False				
ValueRank	-1 (-1 = Scalar)				
DataType	SubscriptionDiagnosticsDataType				
References	Node Class	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the BaseDataVariableType defined in 7.4.					
HasComponent	Variable	SessionId	NodeId	BaseDataVariableType	Mandatory
HasComponent	Variable	SubscriptionId	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	Priority	Byte	BaseDataVariableType	Mandatory
HasComponent	Variable	PublishingInterval	Duration	BaseDataVariableType	Mandatory
HasComponent	Variable	MaxKeepAliveCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	MaxLifetimeCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	MaxNotificationsPerPublish	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	PublishingEnabled	Boolean	BaseDataVariableType	Mandatory
HasComponent	Variable	ModifyCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	EnableCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	DisableCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	RepublishRequestCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	RepublishMessageRequestCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	RepublishMessageCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	TransferRequestCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	TransferredToAltClientCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	TransferredToSameClientCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	PublishRequestCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	DataChangeNotificationsCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	EventNotificationsCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	NotificationsCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	LatePublishRequestCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	CurrentKeepAliveCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	CurrentLifetimeCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	UnacknowledgedMessageCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	DiscardedMessageCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	MonitoredItemCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	DisabledMonitoredItemCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	MonitoringQueueOverflowCount	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	NextSequenceNumber	UInt32	BaseDataVariableType	Mandatory
HasComponent	Variable	EventQueueOverflowCount	UInt32	BaseDataVariableType	Mandatory

7.15 SessionDiagnosticsArrayType

This complex *VariableType* is used for diagnostic information. For each entry of the array instances of this type will provide a *Variable* of the SessionDiagnosticsVariableType *VariableType*, having the SessionDiagnostics as *BrowseName*. Those *Variables* will also be referenced by the SessionDiagnostics *Objects* defined by their type in 6.3.5. The *VariableType* is formally defined in Table 67.

Table 67 – SessionDiagnosticsArrayType Definition

Attribute	Value			
BrowseName	SessionDiagnosticsArrayType			
IsAbstract	False			
ValueRank	1 (1 = OneDimension)			
ArrayDimensions	{0} (0 = UnknownSize)			
DataType	SessionDiagnosticsDataType			
References	NodeClass	BrowseName	DataType TypeDefinition	ModellingRule
Subtype of the BaseDataVariableType defined in 7.4.				
HasComponent	Variable	SessionDiagnostics	SessionDiagnosticsDataType SessionDiagnosticsVariableType	ExposesItsArray

7.16 SessionDiagnosticsVariableType

This complex *VariableType* is used for diagnostic information. Its *DataVariables* reflect its *DataTypes*, having the same semantic defined in 12.11. The *VariableType* is formally defined in Table 68.

Withdrawn

Table 68 – SessionDiagnosticsVariableType Definition

Attribute	Value			
BrowseName	SessionDiagnosticsVariableType			
IsAbstract	False			
ValueRank	-1 (-1 = Scalar)			
Data Type	SessionDiagnosticsDataType			
References	Node Class	BrowseName	Data Type TypeDefinition	Modelling Rule
Subtype of the BaseDataVariableType defined in 7.4.				
HasComponent	Variable	SessionId	NodeId BaseDataVariableType	Mandatory
HasComponent	Variable	SessionName	String BaseDataVariableType	Mandatory
HasComponent	Variable	ClientDescription	ApplicationDescription BaseDataVariableType	Mandatory
HasComponent	Variable	ServerUri	String BaseDataVariableType	Mandatory
HasComponent	Variable	EndpointUrl	String BaseDataVariableType	Mandatory
HasComponent	Variable	LocaleIds	LocaleId[] BaseDataVariableType	Mandatory
HasComponent	Variable	MaxResponseMessageSize	UInt32 BaseDataVariableType	Mandatory
HasComponent	Variable	ActualSessionTimeout	Duration BaseDataVariableType	Mandatory
HasComponent	Variable	ClientConnectionTime	UtcTime BaseDataVariableType	Mandatory
HasComponent	Variable	ClientLastContactTime	UtcTime BaseDataVariableType	Mandatory
HasComponent	Variable	CurrentSubscriptionsCount	UInt32 BaseDataVariableType	Mandatory
HasComponent	Variable	CurrentMonitoredItemsCount	UInt32 BaseDataVariableType	Mandatory
HasComponent	Variable	CurrentPublishRequestsInQueue	UInt32 BaseDataVariableType	Mandatory
HasComponent	Variable	TotalRequestsCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	UnauthorizedRequestsCount	UInt32 BaseDataVariableType	Mandatory
HasComponent	Variable	ReadCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	HistoryReadCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	WriteCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	HistoryUpdateCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	CallCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	CreateMonitoredItemsCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	ModifyMonitoredItemsCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	SetMonitoringModeCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	SetTriggeringCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	DeleteMonitoredItemsCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	CreateSubscriptionCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	ModifySubscriptionCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	SetPublishingModeCount	ServiceCounterDataType BaseDataVariableType	Mandatory
HasComponent	Variable	PublishCount	ServiceCounterDataType	Mandatory