

DIN EN 62541-5

ICS 35.240.50; 35.200

Einsprüche bis 2014-12-24
Vorgesehen als Ersatz für
DIN EN 62541-5:2012-06**Entwurf****OPC Unified Architecture –
Teil 5: Informationsmodell
(IEC 65E/376/CDV:2014);
Englische Fassung FprEN 62541-5:2014**OPC Unified Architecture –
Part 5: Information Model
(IEC 65E/376/CDV:2014);
English version FprEN 62541-5:2014Architecture unifiée OPC –
Partie 5: Modèle d'informations
(CEI 65E/376/CDV:2014);
Version anglaise FprEN 62541-5:2014**Anwendungswarnvermerk**

Dieser Norm-Entwurf mit Erscheinungsdatum 2014-10-24 wird der Öffentlichkeit zur Prüfung und Stellungnahme vorgelegt.

Weil die beabsichtigte Norm von der vorliegenden Fassung abweichen kann, ist die Anwendung dieses Entwurfes besonders zu vereinbaren.

Stellungnahmen werden erbeten

- vorzugsweise online im Norm-Entwurfs-Portal des DIN unter www.entwuerfe.din.de bzw. für Norm-Entwürfe der DKE auch im Norm-Entwurfs-Portal der DKE unter www.entwuerfe.normenbibliothek.de, sofern dort wiedergegeben;
- oder als Datei per E-Mail an dke@vde.com möglichst in Form einer Tabelle. Die Vorlage dieser Tabelle kann im Internet unter www.din.de/stellungnahme oder für Stellungnahmen zu Norm-Entwürfen der DKE unter www.dke.de/stellungnahme abgerufen werden;
- oder in Papierform an die DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE, Stresemannallee 15, 60596 Frankfurt am Main.

Die Empfänger dieses Norm-Entwurfs werden gebeten, mit ihren Kommentaren jegliche relevanten Patentrechte, die sie kennen, mitzuteilen und unterstützende Dokumentationen zur Verfügung zu stellen.

Gesamtumfang 127 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE

Nationales Vorwort

Die Englische Fassung des europäischen Dokuments FprEN 62541-5:2014 „OPC Unified Architecture – Teil 5: Informationsmodell“ (Entwurf in der Umfrage) ist unverändert in diesen Norm-Entwurf übernommen worden.

Die Internationale Elektrotechnische Kommission (IEC) und das Europäische Komitee für Elektrotechnische Normung (CENELEC) haben vereinbart, dass ein auf IEC-Ebene erarbeiteter Entwurf für eine Internationale Norm zeitgleich (parallel) bei IEC und CENELEC zur Umfrage (CDV-Stadium) und Abstimmung als FDIS (en: Final Draft International Standard) bzw. Schluss-Entwurf für eine Europäische Norm gestellt wird, um eine Beschleunigung und Straffung der Normungsarbeit zu erreichen. Dem entsprechend ist das internationale Dokument IEC 65E/376/CDV:2014 „OPC Unified Architecture – Part 5: Information Model“ unverändert in den Entwurf FprEN 62541-5:2014 übernommen worden.

Das internationale Dokument wurde vom SC 65E „Devices and integration in enterprise systems“ der Internationalen Elektrotechnischen Kommission (IEC) erarbeitet.

Bei der Abstimmung zu dem Europäischen Schluss-Entwurf bei CENELEC und dem Internationalen Schluss-Entwurf bei IEC [Final Draft International Standard (FDIS)] sind jeweils nur „JA/NEIN“-Entscheidungen möglich, wobei „NEIN“-Entscheidungen fundiert begründet werden müssen. Dokumente, die bei CENELEC als Europäische Norm angenommen und ratifiziert werden, sind unverändert als Deutsche Normen zu übernehmen.

Für dieses Dokument ist das nationale Arbeitsgremium K 931 „Systemaspekte“ der DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE (www.dke.de) zuständig.

Für den Fall einer undatierten Verweisung im normativen Text (Verweisung auf eine Norm ohne Angabe des Ausgabedatums und ohne Hinweis auf eine Abschnittsnummer, eine Tabelle, ein Bild usw.) bezieht sich die Verweisung auf die jeweils neueste gültige Ausgabe der in Bezug genommenen Norm.

Für den Fall einer datierten Verweisung im normativen Text bezieht sich die Verweisung immer auf die in Bezug genommene Ausgabe der Norm.

Der Zusammenhang der zitierten Normen mit den entsprechenden Deutschen Normen ergibt sich, soweit ein Zusammenhang besteht, grundsätzlich über die Nummer der entsprechenden IEC-Publikation. Beispiel: IEC 60068 ist als EN 60068 als Europäische Norm durch CENELEC übernommen und als DIN EN 60068 ins Deutsche Normenwerk aufgenommen.

Das Präsidium des DIN hat mit Präsidialbeschluss 1/2004 festgelegt, dass DIN-Normen, deren Inhalt sich auf internationale Arbeitsergebnisse der Informationsverarbeitung gründet, unter bestimmten Bedingungen allein in englischer Sprache veröffentlicht werden dürfen. Diese Bedingungen sind für den vorliegenden Normentwurf erfüllt.

Da sich die Benutzer des vorliegenden Normentwurfs der englischen Sprache als Fachsprache bedienen, wird die Englische Fassung der EN 62541-5 veröffentlicht. Zu deren Abschnitt 3, Begriffe, wurde eine Übersetzung angefertigt und als informativer Nationaler Anhang NA des vorliegenden Normentwurfs hinzugefügt. Dieser Normentwurf steht nicht in unmittelbarem Zusammenhang mit Rechtsvorschriften und ist nicht als Sicherheitsnorm anzusehen.

Es ist vorgesehen, auch bei der entsprechenden zukünftigen Deutschen Norm auf die deutsche Sprachfassung zu verzichten und diese in der englischsprachigen Fassung zu veröffentlichen.

Es wird auf die Möglichkeit hingewiesen, dass einige Elemente dieses Dokuments Patentrechte berühren können. ISO [und/oder] IEC sind nicht dafür verantwortlich, einige oder alle diesbezüglichen Patentrechte zu identifizieren.

Das Original-Dokument enthält Bilder in Farbe, die in der Papierversion in einer Graustufen-Darstellung wiedergegeben werden. Elektronische Versionen dieses Dokuments enthalten die Bilder in der originalen Farbdarstellung.

Änderungen

Gegenüber DIN EN 62541-5:2012-06 wurden folgende Änderungen vorgenommen:

- a) In 6.4.35 `ProgressEventType` festgelegt, der den Fortschritt einer Operation wie bei einem Dienstaufwurf feststellt;
- b) in 12.18 einen Datentyp `BitFieldMaskDataType` festgelegt, der ein bit Feld darstellt, in dem einzelne Felder erneut beschrieben werden können, ohne die anderen Felder erneut festzulegen;
- c) in 12.9 im `ServerDiagnosticSummaryDataType` die Eigenschaft `SamplingRateCount` gelöscht, da sie nicht benötigt wurde;
- d) in B.4.4 zu `TransitionVariableType` die Eigenschaft `EffectiveTransitionTime` hinzugefügt;
- e) in 7.19 den variablen Typ `OptionSetType` eingeführt, der eine bit Maske darstellt und Text, der den Sinn der einzelnen bits festlegt;
- f) in 6.4.30 einen neuen `EventType` hinzugefügt `SystemStatusChangeEvent`, der verwendet werden kann, um den Verlust zu einem unterlagerten System anzuzeigen;
- g) in 6.3.2 zu `ServerCapabilitiesTypes` Eigenschaften hinzugefügt, die die größte Länge von arrays und strings beschreibt für Variablen und auch ein Objekt hinzugefügt für die Begrenzung von Operationen (größte arrays, wenn ein Service aufgerufen wird, zum Beispiel Lesen). In 6.3.11 den Typ `OperationLimitsType` hinzugefügt, der diese Informationen beinhaltet;
- h) in 6.4.10 zu `AuditActivateSessionEventType` `SecureChannelId` hinzugefügt und den Text an mehreren Stellen angepasst;
- i) Normativen Anhang C hinzugefügt, der den `FileType` und Methoden zur Übertragung von files festlegt;
- j) in 6.3.1 die Methode `GetMonitoredItems` on `ServerType` hinzugefügt, um Informationen über die beobachteten Gegenstände zu erhalten;
- k) die Vorstellung des `ModelParent` entfernt, da sie nicht nützlich war. Die `NodeId` des `ReferenceType` bleibt erhalten, um bestehende Anwendungen nicht zu beeinträchtigen;
- l) Metadaten für den Namensraum im `ServerType` in 6.3.1 hinzugefügt und Typen für deren Handhabung neu aufgestellt;
- m) in 8.4.4 Darstellung für `ModellingRules` in `OptionalPlaceholder` und `MandatoryPlaceholder` hinzugefügt;
- n) neue Datentypen `NonTransparentNetworkRedundancyType` in 6.3.10, `NetworkGroupDataType` in 12.19 und `EndpointUrlListDataType` in 12.20 hinzugefügt, um die verschiedenen Redundanzarten handhaben zu können. Die Beschreibung der Redundanz erweitert und in 12.5 die Aufzählung `RedundancySupport` auf den neuesten Stand gebracht.

Nationaler Anhang NA (informativ)

3.1 Begriffe

Für die Anwendung dieses Dokuments gelten die Begriffe nach IEC/TR 62541-1 und IEC 62541-3 sowie die folgenden Begriffe.

3.1.1

ClientUserId

(en: **ClientUserId**)

Zeichenkette, die den Benutzer des Clients identifiziert, der eine Aktion anfordert

ANMERKUNG Die `ClientUserId` wird direkt oder indirekt vom `UserIdentityToken` erhalten, der vom Client beim Aufruf des `ActivateSession`-Dienstes übergeben wird. Einzelheiten siehe 6.4.3.

– Leerseite –

CONTENTS

Page

1	Scope.....	1
2	Normative references	1
3	Terms, definitions and conventions.....	1
4	3.1 Terms and definitions	1
5	3.1.1 ClientUserId	1
6	3.2 Abbreviations and symbols	1
7	3.3 Conventions for Node descriptions	2
8	4 NodeIds and BrowseNames.....	4
9	4.1 NodeIds	4
10	4.2 BrowseNames	4
11	5 Common Attributes	4
12	5.1 General	4
13	5.2 Objects.....	5
14	5.3 Variables	5
15	5.4 VariableTypes	5
16	6 Standard ObjectTypes	6
17	6.1 General	6
18	6.2 BaseObjectType	6
19	6.3 ObjectTypes for the Server Object.....	7
20	6.3.1 ServerType.....	7
21	6.3.2 ServerCapabilitiesType.....	9
22	6.3.3 ServerDiagnosticsType	11
23	6.3.4 SessionsDiagnosticsSummaryType	12
24	6.3.5 SessionDiagnosticsObjectType.....	12
25	6.3.6 VendorServerInfoType	13
26	6.3.7 ServerRedundancyType	13
27	6.3.8 TransparentRedundancyType	13
28	6.3.9 NonTransparentRedundancyType	14
29	6.3.10 NonTransparentNetworkRedundancyType	14
30	6.3.11 OperationLimitsType.....	15
31	6.3.12 AddressSpaceFileType	16
32	6.3.13 NamespaceMetadataType	17
33	6.3.14 NamespacesType	18
34	6.4 ObjectTypes used as EventTypes.....	18
35	6.4.1 General	18
36	6.4.2 BaseEventType	19
37	6.4.3 AuditEventType	21
38	6.4.4 AuditSecurityEventType	22
39	6.4.5 AuditChannelEventType	22
40	6.4.6 AuditOpenSecureChannelEventType	23
41	6.4.7 AuditSessionEventType	23
42	6.4.8 AuditCreateSessionEventType	24
43	6.4.9 AuditUrlMismatchEventType	25
44	6.4.10 AuditActivateSessionEventType	25
45	6.4.11 AuditCancelEventType	26

50	6.4.12	AuditCertificateEventType	26
51	6.4.13	AuditCertificateDataMismatchEventType	27
52	6.4.14	AuditCertificateExpiredEventType	27
53	6.4.15	AuditCertificateInvalidEventType	27
54	6.4.16	AuditCertificateUntrustedEventType	28
55	6.4.17	AuditCertificateRevokedEventType	28
56	6.4.18	AuditCertificateMismatchEventType	28
57	6.4.19	AuditNodeManagementEventType	29
58	6.4.20	AuditAddNodesEventType	29
59	6.4.21	AuditDeleteNodesEventType	29
60	6.4.22	AuditAddReferencesEventType	30
61	6.4.23	AuditDeleteReferencesEventType	30
62	6.4.24	AuditUpdateEventType	31
63	6.4.25	AuditWriteUpdateEventType	31
64	6.4.26	AuditHistoryUpdateEventType	32
65	6.4.27	AuditUpdateMethodEventType	32
66	6.4.28	SystemEventType	33
67	6.4.29	DeviceFailureEventType	33
68	6.4.30	SystemStatusChangeEvent	33
69	6.4.31	BaseModelChangeEvent	34
70	6.4.32	GeneralModelChangeEvent	34
71	6.4.33	SemanticChangeEvent	34
72	6.4.34	EventQueueOverflowEventType	35
73	6.4.35	ProgressEventType	35
74	6.5	ModellingRuleType	36
75	6.6	FolderType	36
76	6.7	DataTypeEncodingType	36
77	6.8	DataTypeSystemType	36
78	6.9	AggregateFunctionType	37
79	7	Standard VariableTypes	38
80	7.1	General	38
81	7.2	BaseVariableType	38
82	7.3	PropertyType	38
83	7.4	BaseDataVariableType	38
84	7.5	ServerVendorCapabilityType	39
85	7.6	DataTypeDictionaryType	39
86	7.7	DataTypeDescriptionType	40
87	7.8	ServerStatusType	40
88	7.9	BuildInfoType	41
89	7.10	ServerDiagnosticsSummaryType	41
90	7.11	SamplingIntervalDiagnosticsArrayType	42
91	7.12	SamplingIntervalDiagnosticsType	42
92	7.13	SubscriptionDiagnosticsArrayType	42
93	7.14	SubscriptionDiagnosticsType	43
94	7.15	SessionDiagnosticsArrayType	44
95	7.16	SessionDiagnosticsVariableType	45
96	7.17	SessionSecurityDiagnosticsArrayType	46
97	7.18	SessionSecurityDiagnosticsType	47
98	7.19	OptionSetType	48

99	8	Standard Objects and their Variables	49
100	8.1	General	49
101	8.2	Objects used to organise the AddressSpace structure	49
102	8.2.1	Overview	49
103	8.2.2	Root	49
104	8.2.3	Views	50
105	8.2.4	Objects	50
106	8.2.5	Types	51
107	8.2.6	ObjectTypes	52
108	8.2.7	VariableTypes	53
109	8.2.8	ReferenceTypes	54
110	8.2.9	DataTypes	54
111	8.2.10	OPC Binary	56
112	8.2.11	XML Schema	56
113	8.2.12	EventTypes	56
114	8.3	Server Object and its containing Objects	57
115	8.3.1	General	57
116	8.3.2	Server Object	59
117	8.4	ModellingRule Objects	59
118	8.4.1	ExposesItsArray	59
119	8.4.2	Mandatory	59
120	8.4.3	Optional	60
121	8.4.4	OptionalPlaceholder	60
122	8.4.5	MandatoryPlaceholder	60
123	9	Standard Methods	60
124	9.1	GetMonitoredItems	60
125	10	Standard Views	61
126	11	Standard ReferenceTypes	61
127	11.1	References	61
128	11.2	HierarchicalReferences	61
129	11.3	NonHierarchicalReferences	62
130	11.4	HasChild	62
131	11.5	Aggregates	62
132	11.6	Organizes	63
133	11.7	HasComponent	63
134	11.8	HasOrderedComponent	63
135	11.9	HasProperty	63
136	11.10	HasSubtype	64
137	11.11	HasModellingRule	64
138	11.12	HasTypeDefinition	64
139	11.13	HasEncoding	64
140	11.14	HasDescription	65
141	11.15	HasEventSource	65
142	11.16	HasNotifier	65
143	11.17	GeneratesEvent	65
144	11.18	AlwaysGeneratesEvent	66
145	12	Standard DataTypes	66
146	12.1	Overview	66

147	12.2	DataTypes defined in Part 3	67
148	12.3	DataTypes defined in Part 4	70
149	12.4	BuildInfo	72
150	12.5	RedundancySupport	72
151	12.6	ServerState	73
152	12.7	RedundantServerDataType	73
153	12.8	SamplingIntervalDiagnosticsDataType	74
154	12.9	ServerDiagnosticsSummaryDataType	74
155	12.10	ServerStatusDataType	75
156	12.11	SessionDiagnosticsDataType	75
157	12.12	SessionSecurityDiagnosticsDataType	77
158	12.13	ServiceCounterDataType	77
159	12.14	StatusResult	78
160	12.15	SubscriptionDiagnosticsDataType	79
161	12.16	ModelChangeStructureDataType	80
162	12.17	SemanticChangeStructureDataType	80
163	12.18	BitFieldMaskDataType	81
164	12.19	NetworkGroupDataType	81
165	12.20	EndpointUrlListDataType	81
166	Annex A	(informative) Design decisions when modelling the server information	83
167	A.1	Overview	83
168	A.2	ServerType and Server Object	83
169	A.3	Typed complex Objects beneath the Server Object	83
170	A.4	Properties versus DataVariables	83
171	A.5	Complex Variables using complex DataTypes	84
172	A.6	Complex Variables having an array	84
173	A.7	Redundant information	84
174	A.8	Usage of the BaseDataVariableType	85
175	A.9	Subtyping	85
176	A.10	Extensibility mechanism	85
177	Annex B	(normative) StateMachines	86
178	B.1	General	86
179	B.2	Examples of finite state machines	86
180	B.2.1	Simple state machine	86
181	B.2.2	State machine containing substates	87
182	B.3	Definition of state machine	88
183	B.4	Representation of state machines in the AddressSpace	88
184	B.4.1	Overview	88
185	B.4.2	StateMachineType	89
186	B.4.3	StateVariableType	90
187	B.4.4	TransitionVariableType	91
188	B.4.5	FiniteStateMachineType	91
189	B.4.6	FiniteStateVariableType	92
190	B.4.7	FiniteTransitionVariableType	93
191	B.4.8	StateType	93
192	B.4.9	InitialStateType	94
193	B.4.10	TransitionType	95
194	B.4.11	FromState	95
195	B.4.12	ToState	95

196	B.4.13 HasCause.....	96
197	B.4.14 HasEffect	96
198	B.4.15 HasSubStateMachine	97
199	B.4.16 TransitionEventType.....	97
200	B.4.17 AuditUpdateStateEventType	98
201	B.4.18 Special Restrictions on subtyping StateMachines	98
202	B.4.19 Specific StatusCodes for StateMachines	98
203	B.5 Examples of StateMachines in the AddressSpace	99
204	B.5.1 StateMachineType using inheritance	99
205	B.5.2 StateMachineType with a sub-machine using inheritance	100
206	B.5.3 StateMachineType using containment.....	101
207	B.5.4 Example of a StateMachine having Transition to SubStateMachine	102
208	Annex C (normative) File Transfer	104
209	C.1 Overview	104
210	C.2 FileType	104
211	C.3 Open	105
212	C.4 Close	106
213	C.5 Read	106
214	C.6 Write	107
215	C.7 GetPosition	107
216	C.8 SetPosition.....	108
217	Annex D (informative): Revision 1.02 Highlights.....	109
218		
219		

220

221

222

FIGURES

223	Figure 1 – Standard AddressSpace Structure	49
224	Figure 2 – Views Organization	50
225	Figure 3 – Objects Organization.....	51
226	Figure 4 – ObjectTypes Organization	52
227	Figure 5 – VariableTypes Organization	53
228	Figure 6 – ReferenceType Definitions	54
229	Figure 7 – DataTypes Organization.....	55
230	Figure 8 – EventTypes Organization	56
231	Figure 9 – Excerpt of Diagnostic Information of the Server.....	58
232	Figure B.1 – Example of a simple state machine	87
233	Figure B.2 – Example of a state machine having a sub-machine	87
234	Figure B.3 – The StateMachine Information Model	89
235	Figure B.4 – Example of an initial State in a sub-machine	94
236	Figure B.5 – Example of a StateMachineType using inheritance	99
237	Figure B.6 – Example of a StateMachineType with a SubStateMachine using	
238	inheritance	100
239	Figure B.7 – Example of a StateMachineType using containment.....	101
240	Figure B.8 – Example of a state machine with transitions from sub-states.....	102
241	Figure B.9 – Example of a StateMachineType having Transition to SubStateMachine	103

242

243

244	TABLES	
245	Table 1 – Examples of DataTypes.....	2
246	Table 2 – Type Definition Table	3
247	Table 3 – Common Node Attributes	4
248	Table 4 – Common Object Attributes.....	5
249	Table 5 – Common Variable Attributes	5
250	Table 6 – Common VariableType Attributes	5
251	Table 7 – BaseObjectType Definition	6
252	Table 8 – ServerType Definition.....	7
253	Table 9 – ServerCapabilitiesType Definition.....	9
254	Table 10 – ServerDiagnosticsType Definition	11
255	Table 11 – SessionsDiagnosticsSummaryType Definition	12
256	Table 12 – SessionDiagnosticsObjectType Definition.....	12
257	Table 13 – VendorServerInfoType Definition.....	13
258	Table 14 – ServerRedundancyType Definition.....	13
259	Table 15 – TransparentRedundancyType Definition	13
260	Table 16 – NonTransparentRedundancyType Definition.....	14
261	Table 17 – NonTransparentNetworkRedundancyType Definition	14
262	Table 18 – OperationLimitsType Definition.....	15
263	Table 19 – AddressSpaceFileType Definition	16
264	Table 20 – NamespaceMetadataType Definition	17
265	Table 21 – NamespacesType Definition	18
266	Table 22 – BaseEventType Definition	19
267	Table 23 – AuditEventType Definition	21
268	Table 24 – AuditSecurityEventType Definition.....	22
269	Table 25 – AuditChannelEventType Definition	22
270	Table 26 – AuditOpenSecureChannelEventType Definition	23
271	Table 27 – AuditSessionEventType Definition	23
272	Table 28 – AuditCreateSessionEventType Definition.....	24
273	Table 29 – AuditUrlMismatchEventType Definition	25
274	Table 30 – AuditActivateSessionEventType Definition.....	25
275	Table 31 – AuditCancelEventType Definition	26
276	Table 32 – AuditCertificateEventType Definition	26
277	Table 33 – AuditCertificateDataMismatchEventType Definition	27
278	Table 34 – AuditCertificateExpiredEventType Definition.....	27
279	Table 35 – AuditCertificateInvalidEventType Definition	27
280	Table 36 – AuditCertificateUntrustedEventType Definition	28
281	Table 37 – AuditCertificateRevokedEventType Definition	28
282	Table 38 – AuditCertificateMismatchEventType Definition.....	28
283	Table 39 – AuditNodeManagementEventType Definition	29
284	Table 40 – AuditAddNodesEventType Definition	29
285	Table 41 – AuditDeleteNodesEventType Definition	29
286	Table 42 – AuditAddReferencesEventType Definition.....	30

287	Table 43 – AuditDeleteReferencesEventType Definition.....	30
288	Table 44 – AuditUpdateEventType Definition	31
289	Table 45 – AuditWriteUpdateEventType Definition	31
290	Table 46 – AuditHistoryUpdateEventType Definition	32
291	Table 47 – AuditUpdateMethodEventType Definition.....	32
292	Table 48 – SystemEventType Definition	33
293	Table 49 – DeviceFailureEventType Definition	33
294	Table 50 – SystemStatusChangeEventDefinition	33
295	Table 51 – BaseModelChangeEventDefinition	34
296	Table 52 – GeneralModelChangeEventDefinition.....	34
297	Table 53 – SemanticChangeEventDefinition	34
298	Table 54 – EventQueueOverflowEventType Definition	35
299	Table 55 – ProgressEventType Definition	35
300	Table 56 – ModellingRuleType Definition	36
301	Table 57 – FolderType Definition	36
302	Table 58 – DataTypeEncodingType Definition.....	36
303	Table 59 – DataTypeSystemType Definition.....	36
304	Table 60 – AggregateFunctionType Definition.....	37
305	Table 61 – BaseVariableType Definition	38
306	Table 62 – PropertyType Definition	38
307	Table 63 – BaseDataVariableType Definition	39
308	Table 64 – ServerVendorCapabilityType Definition	39
309	Table 65 – DataTypeDictionaryType Definition.....	39
310	Table 66 – DataTypeDescriptionType Definition.....	40
311	Table 67 – ServerStatusType Definition	40
312	Table 68 – BuildInfoType Definition	41
313	Table 69 – ServerDiagnosticsSummaryType Definition	41
314	Table 70 – SamplingIntervalDiagnosticsArrayType Definition	42
315	Table 71 – SamplingIntervalDiagnosticsType Definition	42
316	Table 72 – SubscriptionDiagnosticsArrayType Definition.....	42
317	Table 73 – SubscriptionDiagnosticsType Definition	43
318	Table 74 – SessionDiagnosticsArrayType Definition	44
319	Table 75 – SessionDiagnosticsVariableType Definition	45
320	Table 76 – SessionSecurityDiagnosticsArrayType Definition	46
321	Table 77 – SessionSecurityDiagnosticsType Definition	47
322	Table 78 – OptionSetType Definition.....	48
323	Table 79 – Root Definition	49
324	Table 80 – Views Definition	50
325	Table 81 – Objects Definition	51
326	Table 82 – Types Definition	51
327	Table 83 – ObjectTypes Definition	52
328	Table 84 – VariableTypes Definition.....	53
329	Table 85 – ReferenceTypes Definition	54

330	Table 86 – DataTypes Definition	55
331	Table 87 – OPC Binary Definition	56
332	Table 88 – XML Schema Definition	56
333	Table 89 – EventTypes Definition	57
334	Table 90 – Server Definition	59
335	Table 91 – ExposesItsArray Definition	59
336	Table 92 – Mandatory Definition	59
337	Table 93 – Optional Definition	60
338	Table 94 – OptionalPlaceholder Definition	60
339	Table 95 – MandatoryPlaceholder Definition	60
340	Table 96 – GetMonitoredItems Method AddressSpace Definition	61
341	Table 97 – References ReferenceType	61
342	Table 98 – HierarchicalReferences ReferenceType	61
343	Table 99 – NonHierarchicalReferences ReferenceType	62
344	Table 100 – HasChild ReferenceType	62
345	Table 101 – Aggregates ReferenceType	62
346	Table 102 – Organizes ReferenceType	63
347	Table 103 – HasComponent ReferenceType	63
348	Table 104 – HasOrderedComponent ReferenceType	63
349	Table 105 – HasProperty ReferenceType	63
350	Table 106 – HasSubtype ReferenceType	64
351	Table 107 – HasModellingRule ReferenceType	64
352	Table 108 – HasTypeDefinition ReferenceType	64
353	Table 109 – HasEncoding ReferenceType	64
354	Table 110 – HasDescription ReferenceType	65
355	Table 111 – HasEventSource ReferenceType	65
356	Table 112 – HasNotifier ReferenceType	65
357	Table 113 – GeneratesEvent ReferenceType	65
358	Table 114 – AlwaysGeneratesEvent ReferenceType	66
359	Table 115 – Part 3 DataType Definitions	67
360	Table 116 – BaseDataType Definition	68
361	Table 117 – Structure Definition	68
362	Table 118 – Enumeration Definition	69
363	Table 119 – ByteString Definition	69
364	Table 120 – Number Definition	69
365	Table 121 – Double Definition	69
366	Table 122 – Integer Definition	69
367	Table 123 – DateTime Definition	70
368	Table 124 – String Definition	70
369	Table 125 – UInteger Definition	70
370	Table 126 – Image Definition	70
371	Table 127 – UInt64 Definition	70
372	Table 128 – Part 4 DataType Definitions	71

373	Table 129 – UserIdentityToken Definition.....	71
374	Table 130 – BuildInfo Structure.....	72
375	Table 131 – BuildInfo Definition	72
376	Table 132 – RedundancySupport Values	72
377	Table 133 – RedundancySupport Definition	72
378	Table 134 – ServerState Values	73
379	Table 135 – ServerState Definition	73
380	Table 136 – RedundantServerDataType Structure	73
381	Table 137 – RedundantServerDataType Definition.....	73
382	Table 138 – SamplingIntervalDiagnosticsDataType Structure	74
383	Table 139 – SamplingIntervalDiagnosticsDataType Definition	74
384	Table 140 – ServerDiagnosticsSummaryDataType Structure.....	74
385	Table 141 – ServerDiagnosticsSummaryDataType Definition	75
386	Table 142 – ServerStatusDataType Structure	75
387	Table 143 – ServerStatusDataType Definition.....	75
388	Table 144 – SessionDiagnosticsDataType Structure	75
389	Table 145 – SessionDiagnosticsDataType Definition.....	77
390	Table 146 – SessionSecurityDiagnosticsDataType Structure	77
391	Table 147 – SessionSecurityDiagnosticsDataType Definition	77
392	Table 148 – ServiceCounterDataType Structure	77
393	Table 149 – ServiceCounterDataType Definition	78
394	Table 150 – StatusResult Structure.....	78
395	Table 151 – StatusResult Definition	78
396	Table 152 – SubscriptionDiagnosticsDataType Structure	79
397	Table 153 – SubscriptionDiagnosticsDataType Definition.....	79
398	Table 154 – ModelChangeStructureDataType Structure.....	80
399	Table 155 – ModelChangeStructureDataType Definition	80
400	Table 156 – SemanticChangeStructureDataType Structure.....	80
401	Table 157 – SemanticChangeStructureDataType Definition	81
402	Table 158 – BitFieldMaskDataType Definition	81
403	Table 159 – NetworkGroupDataType Structure	81
404	Table 160 – NetworkGroupDataType Definition.....	81
405	Table 161 – EndpointUrlListDataType Structure.....	81
406	Table 162 – EndpointUrlListDataType Definition	82
407	Table B.1 – StateMachineType Definition.....	90
408	Table B.2 – StateVariableType Definition.....	90
409	Table B.3 – TransitionVariableType Definition.....	91
410	Table B.4 – FiniteStateMachineType Definition	92
411	Table B.5 – FiniteStateVariableType Definition	93
412	Table B.6 – FiniteTransitionVariableType Definition	93
413	Table B.7 – StateType Definition.....	94
414	Table B.8 – InitialStateType Definition	95
415	Table B.9 – TransitionType Definition	95

416	Table B.10 – FromState ReferenceType	95
417	Table B.11 – ToState ReferenceType	96
418	Table B.12 – HasCause ReferenceType	96
419	Table B.13 – HasEffect ReferenceType	97
420	Table B.14 – HasSubStateMachine ReferenceType	97
421	Table B.15 – TransitionEventType	97
422	Table B.16 – AuditUpdateStateEventType	98
423	Table B.17 – Specific StatusCodes for StateMachines	99
424	Table C.18 – FileType.....	104
425	Table 19 – Open Method AddressSpace Definition	106
426	Table 20 – Close Method AddressSpace Definition	106
427	Table 21 – Read Method AddressSpace Definition.....	107
428	Table 22 – Write Method AddressSpace Definition.....	107
429	Table 23 – GetPosition Method AddressSpace Definition	108
430	Table 24 – SetPosition Method AddressSpace Definition	108
431		
432		

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/XX/FDIS	65E/XX/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.

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

- 489 • reconfirmed,
490 • withdrawn,
491 • replaced by a revised edition, or
492 • amended.

493

494 The National Committees are requested to note that for this publication the stability date
495 is 2017.

496 THIS TEXT IS INCLUDED FOR THE INFORMATION OF THE NATIONAL COMMITTEES AND WILL BE
497 DELETED AT THE PUBLICATION STAGE.

498

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.

499

OPC Unified Architecture Specification

Part 5: Information Model

1 Scope

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

2 Normative references

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

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

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

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

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

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

IEC 62541-9, *OPC Unified Architecture – Part 9: Alarms & Conditions*

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

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

IEC 62541-12, *OPC Unified Architecture – Part 12: Discovery*

3 Terms, definitions and conventions

3.1 Terms and definitions

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

3.1.1 ClientUserId

String that identifies the user of the client requesting an action

NOTE The *ClientUserId* is obtained directly or indirectly from the *UserIdentityToken* passed by the *Client* in the *ActivateSession Service* call. See 6.4.3 for details.

3.2 Abbreviations and symbols

UA	Unified Architecture
XML	Extensible Markup Language

3.3 Conventions for Node descriptions

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

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

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

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

Table 2 – Type Definition Table

Attribute	Value				
Attribute name	Attribute value. If it is an optional Attribute that is not set “--“ will be used.				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
<i>ReferenceType</i> name	<i>NodeClass</i> of the <i>TargetNode</i> .	<i>BrowseName</i> of the target <i>Node</i> . If the <i>Reference</i> is to be instantiated by the server, then the value of the target <i>Node</i> 's <i>BrowseName</i> is “--“.	<i>Attributes</i> of the referenced <i>Node</i> , only applicable for <i>Variables</i> and <i>Objects</i> .		Referenced <i>ModellingRule</i> of the referenced <i>Object</i> .
NOTE Notes referencing footnotes of the table content.					

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

4 NodeIds and BrowseNames

4.1 NodeIds

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

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

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

Note that this standard 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 the OPC Foundation but generated by the *Server*.

4.2 BrowseNames

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

5 Common Attributes

5.1 General

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

Table 3 – Common Node Attributes

Attribute	Value
DisplayName	The <i>DisplayName</i> is a <i>LocalizedText</i> . Each server shall provide the <i>DisplayName</i> identical to the <i>BrowseName</i> of the <i>Node</i> for the <i>LocaleId</i> "en". Whether the server provides translated names for other <i>LocaleIds</i> is vendor specific.
Description	Optionally a vendor specific description is provided.
NodeClass	Shall reflect the <i>NodeClass</i> of the <i>Node</i> .
NodeId	The <i>NodeId</i> is described by <i>BrowseNames</i> as defined in 4.1 and defined in IEC 62541-6.
WriteMask	Optionally the <i>WriteMask Attribute</i> can be provided. If the <i>WriteMask Attribute</i> is provided, it shall set all <i>Attributes</i> to not writeable that are not said to be vendor-specific. For example, the <i>Description Attribute</i> may be set to writeable since a <i>Server</i> may provide a server-specific description for the <i>Node</i> . The <i>NodeId</i> shall not be writeable, because it is defined for each <i>Node</i> in this standard.
UserWriteMask	Optionally the <i>UserWriteMask Attribute</i> can be provided. The same rules as for the <i>WriteMask Attribute</i> apply.

605 **5.2 Objects**

606 For all *Objects* specified in this standard, the *Attributes* named in Table 4 shall be set as specified in
 607 Table 4.

608 **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.

610 **5.3 Variables**

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

613 **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</i> Attribute 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</i> Attribute shall be specified in the table defining the <i>Variable</i> .

615 **5.4 VariableTypes**

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

618 **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> Attribute 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 IEC 62541-3 into the *BaseObjectType*, stating that each *Object* of the *Server* will provide a *NodeVersion*.

6.2 BaseObjectType

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

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

Table 7 – BaseObjectType Definition

Attribute		Value			
BrowseName		BaseObjectType			
IsAbstract		False			
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
HasSubtype	ObjectType	ServerType	Defined in 6.3.1		
HasSubtype	ObjectType	ServerCapabilitiesType	Defined in 6.3.2		
HasSubtype	ObjectType	ServerDiagnosticsType	Defined in 6.3.3		
HasSubtype	ObjectType	SessionsDiagnosticsSummaryType	Defined in 6.3.4		
HasSubtype	ObjectType	SessionDiagnosticsObjectType	Defined in 6.3.5		
HasSubtype	ObjectType	VendorServerInfoType	Defined in 6.3.6		
HasSubtype	ObjectType	ServerRedundancyType	Defined in 6.3.7		
HasSubtype	ObjectType	BaseEventType	Defined in 6.4.2		
HasSubtype	ObjectType	ModellingRuleType	Defined in 6.5		
HasSubtype	ObjectType	FolderType	Defined in 6.6		
HasSubtype	ObjectType	DataTypeEncodingType	Defined in 6.7		
HasSubtype	ObjectType	DataTypeSystemType	Defined in 6.8		

642 **6.3 ObjectTypes for the Server Object**643 **6.3.1 ServerType**

644 This *ObjectType* defines the capabilities supported by the OPC UA *Server*. It is formally defined in
 645 Table 8.

646 **Table 8 – ServerType Definition**

Attribute	Value				
BrowseName	ServerType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the BaseObjectType defined in 6.2					
HasProperty	Variable	ServerArray	String[]	PropertyType	Mandatory
HasProperty	Variable	NamespaceArray	String[]	PropertyType	Mandatory
HasComponent	Variable	ServerStatus ¹	ServerStatusDataType	ServerStatusType	Mandatory
HasProperty	Variable	ServiceLevel	Byte	PropertyType	Mandatory
HasProperty	Variable	Auditing	Boolean	PropertyType	Mandatory
HasComponent	Object	ServerCapabilities ¹	-	ServerCapabilitiesType	Mandatory
HasComponent	Object	ServerDiagnostics ¹	-	ServerDiagnosticsType	Mandatory
HasComponent	Object	VendorServerInfo	-	VendorServerInfoType	Mandatory
HasComponent	Object	ServerRedundancy ¹	-	ServerRedundancyType	Mandatory
HasComponent	Object	Namespaces	-	NamespacesType	Optional
HasComponent	Method	GetMonitoredItems	Defined in 9.1		Optional
NOTE Containing <i>Objects</i> and <i>Variables</i> of these <i>Objects</i> and <i>Variables</i> are defined by their <i>BrowseName</i> defined in the corresponding <i>TypeDefinitionNode</i> . The <i>NodeId</i> is defined by the composed symbolic name described in 4.1.					

647

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

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

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

662 *NamespaceArray* defines an array of namespace URIs. This *Variable* is also referred as *namespace*
 663 *table*. The indexes into the *namespace table* are referred to as *NamespaceIndexes*.
 664 *NamespaceIndexes* are used in *NodeIds* in OPC UA *Services*, rather than the longer namespace
 665 URI. Index 0 is reserved for the OPC UA namespace, and index 1 is reserved for the local *Server*.
 666 Clients may read the entire *namespace table* or they may read individual entries in the *namespace*
 667 *table*. The *Server* shall not modify or delete entries of the *namespace table* while any client has an
 668 open session to the *Server*, because clients may cache the *namespace table*. A *Server* may add
 669 entries to the *namespace table* even if clients are connected to the *Server*. It is recommended that
 670 *Servers* not change the indexes of the *namespace table* but only add entries, because the client may
 671 cache *NodeIds* using the indexes. Nevertheless, it might not always be possible for *Servers* to avoid
 672 changing indexes in the *namespace table*. Clients that cache *NamespaceIndexes* of *NodeIds* should
 673 always check when starting a session to verify that the cached *NamespaceIndexes* have not
 674 changed.

- 675 *ServerStatus* contains elements that describe the status of the *Server*. See 12.10 for a description of
676 its elements.
- 677 *ServiceLevel* describes the ability of the *Server* to provide its data to the client. The value range is
678 from 0 to 255, where 0 indicates the worst and 255 indicates the best. The concrete values are
679 vendor-specific. The intent is to provide the clients an indication of availability among redundant
680 *Servers*.
- 681 *Auditing* is a Boolean specifying if the *Server* is currently generating audit events. It is set to TRUE if
682 the *Server* generates audit events, otherwise to false. The *Profiles* defined in IEC 62541-7 specify
683 what kind of audit events are generated by the *Server*.
- 684 *ServerCapabilities* defines the capabilities supported by the OPC UA *Server*. See 6.3.2 for its
685 description.
- 686 *ServerDiagnostics* defines diagnostic information about the OPC UA *Server*. See 6.3.3 for its
687 description.
- 688 *VendorServerInfo* represents the browse entry point for vendor-defined *Server* information. This
689 *Object* is required to be present even if there are no vendor-defined *Objects* beneath it. See 6.3.6 for
690 its description.
- 691 *ServerRedundancy* describes the redundancy capabilities provided by the *Server*. This *Object* is
692 required even if the *Server* does not provide any redundancy support. If the *Server* supports
693 redundancy, then a subtype of *ServerRedundancyType* is used to describe its capabilities.
694 Otherwise, it provides an *Object* of type *ServerRedundancyType* with the *Property*
695 *RedundancySupport* set to none. See 6.3.7 for the description of *ServerRedundancyType*.
- 696 *Namespaces* provides a list of *NamespaceMetadataType Objects* with additional information about
697 the namespaces used in the *Server*. See 6.3.14 for the description of *NamespaceMetadataType*.
- 698 The *GetMonitoredItems Method* is used to identify the *MonitoredItems* of a subscription. It is defined
699 in 9.1; the intended usage is defined in IEC 62541-4.

700 **6.3.2 ServerCapabilitiesType**

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

703 **Table 9 – ServerCapabilitiesType Definition**

Attribute	Value			
BrowseName	ServerCapabilitiesType			
IsAbstract	False			
References	NodeClass	BrowseName	DataType / TypeDefinition	ModellingRule
Subtype of the BaseObjectType defined in 6.2				
HasProperty	Variable	ServerProfileArray	String[] PropertyType	Mandatory
HasProperty	Variable	LocaleIdArray	LocaleId[] PropertyType	Mandatory
HasProperty	Variable	MinSupportedSampleRate	Duration PropertyType	Mandatory
HasProperty	Variable	MaxBrowseContinuationPoints	UInt16 PropertyType	Mandatory
HasProperty	Variable	MaxQueryContinuationPoints	UInt16 PropertyType	Mandatory
HasProperty	Variable	MaxHistoryContinuationPoints	UInt16 PropertyType	Mandatory
HasProperty	Variable	SoftwareCertificates	SignedSoftwareCertificate[] PropertyType	Mandatory
HasProperty	Variable	MaxArrayLength	UInt32 PropertyType	Optional
HasProperty	Variable	MaxStringLength	UInt32 PropertyType	Optional
HasComponent	Object	OperationLimits	-- OperationLimitsType	Optional
HasComponent	Object	ModellingRules	-- FolderType	Mandatory
HasComponent	Object	AggregateFunctions	-- FolderType	Mandatory
HasComponent	Variable	Vendor specific <i>Variables</i> of a subtype of the ServerVendorCapabilityType defined in 7.5		--

704
 705 *ServerProfileArray* lists the *Profiles* that the *Server* supports. See IEC 62541-7 for the definitions of
 706 *Server Profiles*. This list should be limited to the *Profiles* the *Server* supports in its current
 707 configuration.

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

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

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

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

- 725 *MaxHistoryContinuationPoints* is an integer specifying the maximum number of parallel continuation
726 points of the *HistoryRead Services* that the *Server* can support per session. The value specifies the
727 maximum the *Server* can support under normal circumstances, so there is no guarantee the *Server*
728 can always support the maximum. The client should not open more *HistoryRead* calls with open
729 continuation points than exposed in this *Variable*. The value 0 indicates that the *Server* does not
730 restrict the number of parallel continuation points the client should use.
- 731 *SoftwareCertificates* is an array of *SignedSoftwareCertificates* containing all *SoftwareCertificates*
732 supported by the *Server*. A *SoftwareCertificate* identifies capabilities of the *Server*. It contains the list
733 of *Profiles* supported by the *Server*. *Profiles* are described in IEC 62541-7.
- 734 The *MaxArrayLength Property* indicates the maximum length of a one or multidimensional array
735 supported by *Variables* of the *Server*. In a multidimensional array it indicates the overall length. For
736 example, a three-dimensional array of 2x3x10 has the array length of 60. The *Server* might further
737 restrict the length for individual *Variables* without notice to the client. *Servers* may use the *Property*
738 *MaxArrayLength* defined in IEC 62541-3 on individual *DataVariables* to specify the size on individual
739 values. The individual *Property* may have a larger or smaller value than *MaxArrayLength*.
- 740 The *MaxStringLength Property* indicates the maximum length of *Strings* supported by *Variables* of
741 the *Server*. The *Server* might further restrict the *String* length for individual *Variables* without notice
742 to the client. *Servers* may use the *Property MaxStringLength* defined in IEC 62541-3 on individual
743 *DataVariables* to specify the length on individual values. The individual *Property* may have larger or
744 smaller values than *MaxStringLength*.
- 745 *OperationLimits* is an entry point to access information on operation limits of the *Server*, for example
746 the maximum length of an array in a read *Service* call.
- 747 *ModellingRules* is an entry point to browse to all *ModellingRules* supported by the *Server*. All
748 *ModellingRules* supported by the *Server* should be able to be browsed starting from this *Object*.
- 749 *AggregateFunctions* is an entry point to browse to all *AggregateFunctions* supported by the *Server*.
750 All *AggregateFunctions* supported by the *Server* should be able to be browsed starting from this
751 *Object*. *AggregateFunctions* are *Objects* of *AggregateFunctionType*.
- 752 The remaining components of the *ServerCapabilitiesType* define the server-specific capabilities of
753 the *Server*. Each is defined using a *HasComponent Reference* whose target is an instance of a
754 vendor-defined subclass of the abstract *ServerVendorCapabilityType* (see 7.5). Each subtype of this
755 type defines a specific *Server* capability. The *NodeIds* for these *Variables* and their *VariableTypes*
756 are server-defined.

6.3.3 ServerDiagnosticsType

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

Table 10 – ServerDiagnosticsType Definition

Attribute	Value			
BrowseName	ServerDiagnosticsType			
IsAbstract	False			
References	Node Class	BrowseName	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	Modelling Rule
Subtype of the BaseObjectType defined in 6.2				
HasComponent	Variable	SessionDiagnosticsArray	SessionDiagnosticsDataType[] SessionDiagnosticsArrayType	Mandatory
HasComponent	Variable	SessionSecurityDiagnosticsArray	SessionSecurityDiagnosticsDataType[] SessionSecurityDiagnosticsArrayType	Mandatory
HasComponent	Object	<ClientName>	-- SessionDiagnosticsObjectType	Optional Placeholder
NOTE This row represents no <i>Node</i> in the <i>AddressSpace</i> . It is a placeholder pointing out that instances of the <i>ObjectType</i> will have those <i>Objects</i> .				

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

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

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

6.3.5 SessionDiagnosticsObjectType

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

Table 12 – SessionDiagnosticsObjectType Definition

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

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

SubscriptionDiagnosticsArray is an array of Subscription diagnostic information per opened subscription, as defined in 12.15. Its *TypeDefinitionNode* is the *VariableType*.

814 SubscriptionDiagnosticsArrayType providing a *Variable* for each entry in the array, as defined in
815 7.13.

816 6.3.6 VendorServerInfoType

817 This *ObjectType* defines a placeholder *Object* for vendor-specific information about the OPC UA
818 *Server*. This *ObjectType* defines an empty *ObjectType* that has no components. It shall be subtyped
819 by vendors to define their vendor-specific information. This *ObjectType* is formally defined in Table
820 13.

821 Table 13 – VendorServerInfoType Definition

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

822 6.3.7 ServerRedundancyType

824 This *ObjectType* defines the redundancy capabilities supported by the OPC UA *Server*. It is formally
825 defined in Table 14.

826 Table 14 – ServerRedundancyType Definition

Attribute	Value				
BrowseName	ServerRedundancyType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	Type Definition	Modelling Rule
Subtype of the BaseObjectType defined in 6.2					
HasProperty	Variable	RedundancySupport	RedundancySupport	PropertyType	Mandatory
HasSubtype	ObjectType	TransparentRedundancyType	Defined in 6.3.8		
HasSubtype	ObjectType	NonTransparentRedundancyType	Defined in 6.3.9		

827
828 *RedundancySupport* indicates what redundancy is supported by the *Server*. Its values are defined in
829 12.5. It shall be set to NONE_0 for all instances of the *ServerRedundancyType* using the *ObjectType*
830 directly (no subtype).

831 6.3.8 TransparentRedundancyType

832 This *ObjectType* is a subtype of *ServerRedundancyType* and is used to identify the capabilities of
833 the OPC UA *Server* for server-controlled redundancy with a transparent switchover for the client. It is
834 formally defined in Table 15.

835 Table 15 – TransparentRedundancyType Definition

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

836
837 *RedundancySupport* is inherited from the *ServerRedundancyType*. It shall be set to
838 TRANSPARENT_4 for all instances of the *TransparentRedundancyType*.

839 Although, in a transparent switchover scenario, all redundant *Servers* serve under the same URI to
840 the client, it may be required to track the exact data source on the client. Therefore,

contains an identifier of the currently-used *Server* in the redundant set. This *Server* is valid only inside a session; if a client opens several sessions, different *Servers* of the redundant set of *Servers* may serve it in different sessions. The value of the *CurrentServerId* may change due to failover or load balancing, so a client that needs to track its data source shall subscribe to this *Variable*.

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

6.3.9 NonTransparentRedundancyType

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

Table 16 – NonTransparentRedundancyType Definition

Attribute	Value				
BrowseName	NonTransparentRedundancyType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the <i>ServerRedundancyType</i> defined in 6.3.7, which means it inherits the <i>InstanceDeclarations</i> of that Node.					
HasProperty	Variable	ServerUriArray	String[]	PropertyType	Mandatory
HasSubtype	ObjectType	NonTransparentNetworkRedundancyType	Defined in 6.3.10		

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

RedundancySupport is inherited from the *ServerRedundancyType*. It shall be set to COLD_1, WARM_2, HOT_3 or HOT_AND_MIRRORED_5 for all instances of the *NonTransparentRedundancyType*. It defines the redundancy support provided by the *Server*. The *Client* is allowed to access the redundant *Server* only as described there, however, "hot" switchover implies the support of "warm" switchover and "warm" switchover implies the support of "cold" switchover. Support for HotAndMirrored redundancy implies the support of "hot" switchover, however, for *Servers* supporting HotAndMirrored redundancy it is strongly recommended that *Clients* use the HotAndMirrored mechanisms.

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

6.3.10 NonTransparentNetworkRedundancyType

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

Table 17 – NonTransparentNetworkRedundancyType Definition

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

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

881 *RedundancySupport* is inherited from the *ServerRedundancyType*. It shall be set to *COLD_1*,
 882 *WARM_2*, *HOT_3* or *HOT_AND_MIRRORED_5* for all instances of the
 883 *NonTransparentNetworkRedundancyType*. If no server redundancy is supported (the *ServerUriArray*
 884 only contains one entry), the *RedundancySupport* shall be set to *HOT_AND_MIRRORED_5*.

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

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

896 6.3.11 OperationLimitsType

897 This *ObjectType* is a subtype of *FolderType* and is used to identify the operation limits of the OPC
 898 UA *Server*. It is formally defined in Table 18.

899 Table 18 – OperationLimitsType Definition

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

900
 901 The *MaxNodesPerRead Property* indicates the maximum size of the *nodesToRead* array when a
 902 *Client* calls the *Read Service*.

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

906 The *MaxNodesPerHistoryReadEvents Property* indicates the maximum size of the *nodesToRead*
 907 array when a *Client* calls the *HistoryRead Service* using the *historyReadDetails* EVEN

- 908 The *MaxNodesPerWrite Property* indicates the maximum size of the *nodesToWrite* array when a
909 *Client* calls the *Write Service*.
- 910 The *MaxNodesPerHistoryUpdateData Property* indicates the maximum size of the
911 *historyUpdateDetails* array supported by the *Server* when a *Client* calls the *HistoryUpdate Service*
912 using *historyReadDetails* RAW, PROCESSED, MODIFIED or ATTIME.
- 913 The *MaxNodesPerHistoryUpdateEvents Property* indicates the maximum size of the
914 *historyUpdateDetails* array when a *Client* calls the *HistoryUpdate Service* using *historyReadDetails*
915 *EVENTS*.
- 916 The *MaxNodesPerMethodCall Property* indicates the maximum size of the *methodsToCall* array
917 when a *Client* calls the *Call Service*.
- 918 The *MaxNodesPerBrowse Property* indicates the maximum size of the *nodesToBrowse* array when
919 calling the *Browse Service* or the *continuationPoints* array when a *Client* calls the *BrowseNext*
920 *Service*.
- 921 The *MaxNodesPerRegisterNodes Property* indicates the maximum size of the *nodesToRegister* array
922 when a *Client* calls the *RegisterNodes Service* and the maximum size of the *nodesToUnregister*
923 when calling the *UnregisterNodes Service*.
- 924 The *MaxNodesPerTranslateBrowsePathsToNodeIds Property* indicates the maximum size of the
925 *browsePaths* array when a *Client* calls the *TranslateBrowsePathsToNodeIds Service*.
- 926 The *MaxNodesPerNodeManagement Property* indicates the maximum size of the *nodesToAdd* array
927 when a *Client* calls the *AddNodes Service*, the maximum size of the *referencesToAdd* array when a
928 *Client* calls the *AddReferences Service*, the maximum size of the *nodesToDelete* array when a *Client*
929 calls the *DeleteNodes Service*, and the maximum size of the *referencesToDelete* array when a *Client*
930 calls the *DeleteReferences Service*.
- 931 The *MaxMonitoredItemsPerCall Property* indicates the maximum size of the *itemsToCreate* array
932 when a *Client* calls the *CreateMonitoredItems Service*, the maximum size of the *itemsToModify* array
933 when a *Client* calls the *ModifyMonitoredItems Service*, the maximum size of the *monitoredItemIds*
934 array when a *Client* calls the *SetMonitoringMode Service*, and the maximum size of the *linksToAdd*
935 and the *linksToRemove* arrays when a *Client* calls the *SetTriggering Service*.

936 6.3.12 AddressSpaceFileType

- 937 This *ObjectType* defines the file for a namespace provided by the OPC UA *Server*. It is formally
938 defined in Table 19. It represents an XML address space file using the XML schema defined in
939 IEC 62541-6.

940 Table 19 – AddressSpaceFileType Definition

Attribute	Value				
BrowseName	AddressSpaceFileType				
IsAbstract	False				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modelling Rule
Subtype of the <i>FileType</i> defined in C.2					
HasComponent	Method	ExportNamespace	The method has no parameters.		Optional

941

942 The *ExportNamespace Method* provides a way to export the namespace from the *Server*
 943 *AddressSpace* to the XML file represented by the *AddressSpaceFileType*. *Value Attributes* are only
 944 exported if they represent static configuration information. The client is expected to call the
 945 *ExportNamespace Method* first to update the XML file and then access the file with the *Methods*
 946 defined in the *FileType*.

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

949 6.3.13 NamespaceMetadataType

950 This *ObjectType* defines the metadata for a namespace provided by the *Server*. It is formally defined
 951 in Table 20.

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

955 **Table 20 – NamespaceMetadataType Definition**

Attribute	Value				
BrowseName	NamespaceMetadataType				
IsAbstract	False				
References	NodeClass	BrowseName	Data Type	TypeDefinition	Modelling Rule
Subtype of the BaseObjectType defined in 6.2					
HasProperty	Variable	NamespaceUri	String	PropertyType	Mandatory
HasProperty	Variable	NamespaceVersion	String	PropertyType	Mandatory
HasProperty	Variable	NamespacePublicationDate	DateTime	PropertyType	Mandatory
HasProperty	Variable	IsNamespaceSubset	Boolean	PropertyType	Mandatory
HasProperty	Variable	StaticNodeIdTypes	IdType[]	PropertyType	Mandatory
HasProperty	Variable	StaticNumericNodeIdRange	NumericRange[]	PropertyType	Mandatory
HasProperty	Variable	StaticStringNodeIdPattern	String	PropertyType	Mandatory
HasComponent	Object	NamespaceFile	-	AddressSpaceFileType	Optional

956
 957 The *BrowseName* of instances of this type shall be derived from the represented namespace. This
 958 can, for example, be done by using the index of the namespace in the *NamespaceArray* as
 959 *namespaceIndex* of the *QualifiedName* and the namespace URI as *name* of the *QualifiedName*.

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

962 The *NamespaceVersion Property* provides version information for the namespace. It is intended for
 963 display purposes and shall not be used to programmatically identify the latest version.

964 The *NamespacePublicationDate Property* provides the publication date of the namespace version.
 965 This *Property* value can be used by *Clients* to determine the latest version if different versions are
 966 provided by different *Servers*.

967 The *IsNamespaceSubset Property* defines whether all *Nodes* of the namespace are accessible in the
 968 *Server* or only a subset. It is set to FALSE if the full namespace is provided and TRUE if not.

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

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

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

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

984 The *Object NamespaceFile* contains all *Nodes* and *References* of the namespace in an XML file
 985 where the XML schema is defined in IEC 62541-6. The XML file is provided through an
 986 *AddressSpaceFileType Object*.

987 6.3.14 NamespacesType

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

Table 21 – NamespacesType Definition

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

991
 992 The *ObjectType* contains a list of *NamespaceMetadataType Objects* representing the namespaces in
 993 the *Server*. The *BrowseName* of an *Object* shall be derived from the namespace represented by the
 994 *Object*. This can, for example, be done by using the index of the namespace in the *NamespaceArray*
 995 as *namespaceIndex* of the *QualifiedName* and the namespace URI as *name* of the *QualifiedName*.

996 6.4 ObjectTypes used as EventTypes

997 6.4.1 General

998 This International Standard defines standard *EventTypes*. They are represented in the
 999 *AddressSpace* as *ObjectTypes*. The *EventTypes* are already defined in IEC 62541-3. The following
 1000 subclauses specify their representation in the *AddressSpace*.

1001 **6.4.2 BaseEventType**

1002 This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined
1003 in Table 22.

1004 **Table 22 – BaseEventType Definition**

Attribute	Value				
BrowseName	BaseEventType				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the <i>BaseObjectType</i> defined in 6.2					
HasSubtype	ObjectType	AuditEventType	Defined in 6.4.3		
HasSubtype	ObjectType	SystemEventType	Defined in 6.4.28		
HasSubtype	ObjectType	BaseModelChangeEvent	Defined in 6.4.31		
HasSubtype	ObjectType	SemanticChangeEvent	Defined in 6.4.33		
HasSubtype	ObjectType	EventQueueOverflowEvent	Defined in 6.4.34		
HasSubtype	ObjectType	ProgressEvent	Defined in 6.4.35		
HasProperty	Variable	EventId	ByteString	PropertyType	Mandatory
HasProperty	Variable	EventType	NodeId	PropertyType	Mandatory
HasProperty	Variable	SourceNode	NodeId	PropertyType	Mandatory
HasProperty	Variable	SourceName	String	PropertyType	Mandatory
HasProperty	Variable	Time	UtcTime	PropertyType	Mandatory
HasProperty	Variable	ReceiveTime	UtcTime	PropertyType	Mandatory
HasProperty	Variable	LocalTime	TimeZoneDataType	PropertyType	Optional
HasProperty	Variable	Message	LocalizedText	PropertyType	Mandatory
HasProperty	Variable	Severity	UInt16	PropertyType	Mandatory

1005
1006 *EventId* is generated by the *Server* to uniquely identify a particular *Event Notification*. The *Server* is
1007 responsible to ensure that each *Event* has its unique *EventId*. It may do this, for example, by putting
1008 GUIDs into the *ByteString*. Clients can use the *EventId* to assist in minimizing or eliminating gaps
1009 and overlaps that may occur during a redundancy failover. The *EventId* shall always be returned as
1010 value and the *Server* is not allowed to return a *StatusCode* for the *EventId* indicating an error.

1011 *EventType* describes the specific type of *Event*. The *EventType* shall always be returned as value
1012 and the *Server* is not allowed to return a *StatusCode* for the *EventType* indicating an error.

1013 *SourceNode* identifies the *Node* that the *Event* originated from. If the *Event* is not specific to a *Node*
1014 the *NodeId* is set to null. Some subtypes of this *BaseEventType* may define additional rules for
1015 *SourceNode*.

1016 *SourceName* provides a description of the source of the *Event*. This could be the string-part of the
1017 *DisplayName* of the *Event* source using the default locale of the server, if the *Event* is specific to a
1018 *Node*, or some server-specific notation.

1019 *Time* provides the time the *Event* occurred. This value is set as close to the event generator as
1020 possible. It often comes from the underlying system or device. Once set, intermediate OPC UA
1021 *Servers* shall not alter the value.

1022 *ReceiveTime* provides the time the OPC UA *Server* received the *Event* from the underlying device of
1023 another *Server*. *ReceiveTime* is analogous to *ServerTimestamp* defined in IEC 62541-4, i.e. in the
1024 case where the OPC UA *Server* gets an *Event* from another OPC UA *Server*, each *Server* applies its
1025 own *ReceiveTime*. That implies that a *Client* may get the same *Event*, having the same *EventId*, from
1026 different *Servers* having different values of the *ReceiveTime*. The *ReceiveTime* shall always be
1027 returned as value and the *Server* is not allowed to return a *StatusCode* for the *ReceiveTime*
1028 indicating an error.

1029 *LocalTime* is a structure containing the *Offset* and the *DaylightSavingInOffset* flag. The *Offset*
1030 specifies the time difference (in minutes) between the *Time Property* and the time at the location in
1031 which the event was issued. If *DaylightSavingInOffset* is TRUE, then Standard/Daylight savings time

1032 (DST) at the originating location is in effect and Offset includes the DST correction. If FALSE then
1033 the Offset does not include DST correction and DST may or may not have been in effect.

1034 *Message* provides a human-readable and localizable text description of the *Event*. The *Server* may
1035 return any appropriate text to describe the *Event*. A null string is not a valid value; if the *Server* does
1036 not have a description, it shall return the string part of the *BrowseName* of the *Node* associated with
1037 the *Event*.

1038 *Severity* is an indication of the urgency of the *Event*. This is also commonly called “priority”. Values
1039 will range from 1 to 1 000, with 1 being the lowest severity and 1 000 being the highest. Typically, a
1040 severity of 1 would indicate an *Event* which is informational in nature, while a value of 1 000 would
1041 indicate an *Event* of catastrophic nature, which could potentially result in severe financial loss or
1042 loss of life.

1043 It is expected that very few *Server* implementations will support 1 000 distinct severity levels.
1044 Therefore, *Server* developers are responsible for distributing their severity levels across the 1 –
1045 1 000 range in such a manner that clients can assume a linear distribution. For example, a client
1046 wishing to present five severity levels to a user should be able to do the following mapping:

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

1047
1048 In many cases a strict linear mapping of underlying source severities to the OPC Severity range is
1049 not appropriate. The *Server* developer will instead intelligently map the underlying source severities
1050 to the 1 to 1 000 OPC Severity range in some other fashion. In particular, it is recommended that
1051 *Server* developers map *Events* of high urgency into the OPC severity range of 667 to 1 000, *Events*
1052 of medium urgency into the OPC severity range of 334 to 666 and *Events* of low urgency into OPC
1053 severities of 1 to 333.

1054 For example, if a source supports 16 severity levels that are clustered such that severities 0 to 2 are
1055 considered to be LOW, 3 to 7 are MEDIUM and 8 –to 15 are HIGH, then an appropriate mapping might
1056 be as follows:

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

1057
1058 Some *Servers* might not support any *Events* which are catastrophic in nature, so they may choose to
1059 map all of their severities into a subset of the 1 to 1 000 range (for example, 1 to 666). Other

1060 Servers might not support any *Events* which are merely informational, so they may choose to map all
1061 of their severities into a different subset of the 1 to 1 000 range (for example, 334 to 1 000).

1062 The purpose of this approach is to allow clients to use severity values from multiple *Servers* from
1063 different vendors in a consistent manner. Additional discussions of severity can be found in
1064 IEC 62541-9.

1065 6.4.3 AuditEventType

1066 This *EventType* is defined in IEC 62541-3. Its representation in the *AddressSpace* is formally defined
1067 in Table 23.

1068 Table 23 – AuditEventType Definition

Attribute		Value			
BrowseName		AuditEventType			
IsAbstract		True			
References	NodeClass	BrowseName	DataType	TypeDefinition	Modelling Rule
Subtype of the <i>BaseEventType</i> defined in 6.4.2, which means it inherits the InstanceDeclarations of that Node.					
HasSubtype	ObjectType	AuditSecurityEventType	Defined in 6.4.4		
HasSubtype	ObjectType	AuditNodeManagementEventType	Defined in 6.4.19		
HasSubtype	ObjectType	AuditUpdateEventType	Defined in 6.4.24		
HasSubtype	ObjectType	AuditUpdateMethodEventType	Defined in 6.4.27		
HasProperty	Variable	ActionTimeStamp	UtcTime	PropertyType	Mandatory
HasProperty	Variable	Status	Boolean	PropertyType	Mandatory
HasProperty	Variable	ServerId	String	PropertyType	Mandatory
HasProperty	Variable	ClientAuditEntryId	String	PropertyType	Mandatory
HasProperty	Variable	ClientUserId	String	PropertyType	Mandatory

1069
1070 This *EventType* inherits all *Properties* of the *BaseEventType*. Their semantic is defined in 6.4.2.

1071 *ActionTimeStamp* identifies the time the user initiated the action that resulted in the *AuditEvent*
1072 being generated. It differs from the *Time Property* because this is the time the server generated the
1073 *AuditEvent* documenting the action.

1074 *Status* identifies whether the requested action could be performed (set *Status* to TRUE) or not (set
1075 *Status* to FALSE).

1076 *ServerId* uniquely identifies the *Server* generating the *Event*. It identifies the *Server* uniquely even in
1077 a server-controlled transparent redundancy scenario where several *Servers* may use the same URI.

1078 *ClientAuditEntryId* contains the human-readable *AuditEntryId* defined in IEC 62541-3.

1079 The *ClientUserId* identifies the user of the client requesting an action. The *ClientUserId* can be
1080 obtained from the *UserIdentityToken* passed in the *ActivateSession* call. If the *UserIdentityToken* is
1081 a *UserNameIdentityToken* then the *ClientUserId* is the *UserName*. If the *UserIdentityToken* is an
1082 *X509IdentityToken* then the *ClientUserId* is the X509 Subject Name of the *Certificate*. If the
1083 *UserIdentityToken* is an *IssuedIdentityToken* then the *ClientUserId* should be a string that represents
1084 the owner of the token. The best choice for the string depends on the type of *IssuedIdentityToken*. If
1085 an *AnonymousIdentityToken* was used, the value is null.

6.4.4 AuditSecurityEventType

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

Table 24 – AuditSecurityEventType Definition

Attribute	Value				
BrowseName	AuditSecurityEventType				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the AuditEventType defined in 6.4.3, which means it inherits the InstanceDeclarations of that Node.					
HasSubtype	ObjectType	AuditChannelEventType	Defined in 6.4.5		
HasSubtype	ObjectType	AuditSessionEventType	Defined in 6.4.7		
HasSubtype	ObjectType	AuditCertificateEventType	Defined in 6.4.12		

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

6.4.5 AuditChannelEventType

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

Table 25 – AuditChannelEventType Definition

Attribute	Value				
BrowseName	AuditChannelEventType				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the AuditSecurityEventType defined in 6.4.4, which means it inherits the InstanceDeclarations of that Node.					
HasSubtype	ObjectType	AuditOpenSecureChannelEventType	Defined in 6.4.6		
HasProperty	Variable	SecureChannelId	String	PropertyType	Mandatory

This *EventType* inherits all *Properties* of the *AuditSecurityEventType*. Their semantic is defined in 6.4.4. There are no additional *Properties* defined for this *EventType*. The *SourceNode* for *Events* of this type should be assigned to the *Server Object*. The *SourceName* for *Events* of this type should be "SecureChannel/" and the *Service* that generates the *Event* (e.g. SecureChannel/OpenSecureChannel or SecureChannel/CloseSecureChannel). If the *ClientUserId* is not available for a *CloseSecureChannel* call, then this parameter shall be set to "System/CloseSecureChannel".

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