DIN EN IEC 62541-3

DIN

ICS 25.040.40; 35.200; 35.240.50

Entwurf

Einsprüche bis 2019-10-16
Vorgesehen als Ersatz für
DIN EN 62541-3:2015-11

OPC Unified Architecture – Teil 3: Adressraummodell (IEC 65E/601/CDV:2018); Englische Fassung prEN IEC 62541-3:2018

OPC unified architecture –
Part 3: Address Space Model
(IEC 65E/601/CDV:2018);
English version prEN IEC 62541-3:2018

Architecture unifiée OPC –
Partie 3: Modèle de l'Espace d'Adressage (AdressSpace)
(IEC 65E/601/CDV:2018);
Version anglaise prEN IEC 62541-3:2018

Anwendungswarnvermerk

Dieser Norm-Entwurf mit Erscheinungsdatum 2019-08-16 wird der Öffentlichkeit zur Prüfung und Stellungnahme vorgelegt.

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

Stellungnahmen werden erbeten

- vorzugsweise online im Norm-Entwurfs-Portal von DIN unter www.din.de/go/entwuerfe 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/go/stellungnahmen-norm-entwuerfe 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 in 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 114 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE

AVA Documents
Document Sharing Hub

Full document details: E DIN EN IEC 62541-3:2019-09

Nationales Vorwort

Die Englische Fassung des europäischen Dokuments prEN IEC 62541-3:2018 „OPC Unified Architecture – Teil 3: Adressraummodell“ (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/601/CDV:2018 „OPC unified architecture – Part 3: Address Space Model“ unverändert in den Entwurf prEN IEC 62541-3:2018 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 Automatisierung“ der DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE (www.dke.de) zuständig.

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

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

Der Zusammenhang der zitierten Dokumente mit den entsprechenden deutschen Dokumenten 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 Norm-Entwurf erfüllt.

Da sich die Benutzer des vorliegenden Norm-Entwurfs der englischen Sprache als Fachsprache bedienen, wird die englische Fassung der prEN IEC 62541-3 veröffentlicht. Zu deren Abschnitt 1, der den Anwendungsbereich festlegt, und Unterabschnitt 3.1, der die Begriffe festlegt, wurde eine Übersetzung angefertigt und als informativer Nationaler Anhang NA des vorliegenden Norm-Entwurfs hinzugefügt. Für einige der verwendeten Begriffe existieren keine gebräuchlichen deutschen Benennungen, da sich die deutschen Anwender in der Regel ebenfalls der englischen Benennungen bedienen. Dieser Norm-Entwurf 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. IEC ist nicht dafür verantwortlich, einige oder alle diesbezüglichen Patentrechte zu identifizieren.

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 62541-3:2019-09

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 62541-3:2015-11 wurden folgende Änderungen vorgenommen:

a) neuen verbesserten Ansatz zur Anzeige von Strukturdefinitionen hinzugefügt. Ein Attribut eines DataType Knoten enthält nun einfach eine binäre Beschreibung;

b) neue Flags für Variablen zur Angabe der Atomarität beim Lesen oder Schreiben hinzugefügt.;

c) Rollen und Zulassungen zur Ermöglichung der Konfiguration einer rollenbasierten Autorisierung hinzugefügt;

d) neue Datentypen hinzugefügt: „Union“, „Decimal“, „OptionSet“, „DateString“, „TimeString“, „DurationString“, „NormalizedString“, „DecimalString“ und „AudioDataType“;

e) Definition zur Anwendung der ModellingRules OptionalPlaceHolder und MandatoryPlaceHolder für Verfahren hinzugefügt;

f) optionale Eigenschaften „MaxCharacters“ und „MaxByteStringLength“ zu variablen Knoten hinzugefügt.

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 62541-3:2019-09

Nationaler Anhang NA
(informativ)

Auszugsweise Übersetzung

Reihenfolge und Inhalt der folgenden Abschnitte sind identisch mit denen in den jeweiligen Abschnitten der Englischen Fassung.

1 Anwendungsbereich

Diese Spezifikation legt den *AdressRaum* (en: *AddressSpace*) und seine *Objekte* (en: *Objects*) der OPC Unified Architecture (vereinheitlichte Architektur der plattformunabhängigen Kommunikation) fest. Dieser Teil stellt das OPC-UA-Metamodell dar, auf welchem OPC-UA-Informationsmodelle beruhen.

3 Begriffe, Abkürzungen und Konventionen

3.1 Begriffe

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

ISO und IEC stellen terminologische Datenbanken für die Anwendung in der Normung unter den folgenden Adressen bereit:

- IEC Electropedia: verfügbar unter http://www.electropedia.org/
- ISO Online Browsing Platform: verfügbar unter http://www.iso.org/obp

3.1.1
DatenTyp
(en: DataType)
Instanz eines *DatenTyps Knoten* (en: *Data Type Node*), die gemeinsam mit dem *Attribut* (en: *Attribute*) *WertRang* (en: *ValueRank*) zur Festlegung einer *Variablen* verwendet wird

3.1.2
DatenTypID
(en: DataTypeID)
KnotenID (Identität eines Knoten) (en: NodeID) eines *DatenTyps Knoten*

3.1.3
DatenVariable
(en: DataVariable)
Variablen *(en: Variables), die* Werte *von* Objekten*, entweder direkt oder indirekt für komplexe* Variablen *darstellen; dabei sind* Variablen *stets der* ZielKnoten *(en: TargetNode) einer* Referenz *(en: Reference)* BesitztKompetenz *(en: HasCompetent)*

3.1.4
EreignisTyp
(en: EvenType)
ObjektTyp (en: ObjectType) Knoten der die Typdefinition eines *Ereignisses (en: Event)* darstellt

3.1.5
hierarchische Referenz
(en: Hierarchical Reference)
Referenz, welche zum Aufbau von im Hierarchien im *AdressRaum* (en: *AdressSpace*) verwendet wird

Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

Anmerkung 1 zum Begriff: Alle hierarchischen ReferenzTypen (en: ReferenceTypes) werden von HierarchischenReferenzen (en: HierarchicalReferences) abgeleitet.

3.1.6
InstanzAnzeige
(en: InstanceDeclaration)
Knoten, der von einem komplexen *TypDefinitionsKnoten* (en: TypeDefinitionNode) zum Anzeigen einer komplexen Struktur verwendet wird

Anmerkung 1 zum Begriff: Es ist eine Instanz, die von einer Typdefinition verwendet wird.

3.1.7
ModellierungsRegel
(en: ModellingRule)
Metadaten einer *InstanzAnzeige*, die festlegen, wie die *InstanzAnzeige* für die Instantiierung zu verwenden ist; sie legt auch Untertypenregeln für eine *InstanzAnzeige* fest

3.1.8
Eigenschaft
(en: Property)
Variablen, die der *ZielKnoten* (en: *TargetNode*) für eine *Referenz BesitztEigenschaft* (en: HasProperty) sind

Anmerkung 1 zum Begriff: Eigenschaften beschreiben die Merkmale eines Knoten.

3.1.9
QuellKnoten
(en: SourceNode)
Knoten, der eine *Referenz* zu einem weiteren *Knoten* hat

BEISPIEL: In der *Referenz* „A enthält B“ ist „A“ der *QuellKnoten*.

3.1.10
ZielKnoten
(en: TargetNode)
Knoten, auf den von einem weiteren *Knoten* Bezug genommen wird

BEISPIEL: In der *Referenz* „A enthält B“ ist „B“ der *ZielKnoten*.

3.1.11
TypDefinitionsKnoten
(en: TypeDefinitionNode)
Knoten, der zur Festlegung des Typs eines weiteren *Knoten* verwendet wird

Anmerkung 1 zum Begriff: Die Knoten *ObjektTyp* (en: *ObjectType*) und VariablenTyp sind *TypDefinitionsKnoten*.

3.1.12
VariablenTyp
(en: VariableType)
Knoten, der die Typdefinition für eine *Variable* darstellt

Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

– Leerseite –

Documents
Document Sharing Hub

Full document details: E DIN EN IEC 62541-3:2019-09

CONTENTS

FIGURES ..... - 8 -
TABLES ..... - 10 -
FOREWORD 12
1 Scope 14
2 Normative references 14
3 Terms, definitions, abbreviations and conventions 15
3.1 Terms and definitions 15
3.2 Abbreviations 16
3.3 Conventions 16
3.3.1 Conventions for AddressSpace figures 16
3.3.2 Conventions for defining NodeClasses 16
4 AddressSpace concepts 17
4.1 Overview 17
4.2 Object Model 18
4.3 Node Model 18
4.3.1 General 18
4.3.2 NodeClasses 18
4.3.3 Attributes 19
4.3.4 References 19
4.4 Variables 20
4.4.1 General 20
4.4.2 Properties 20
4.4.3 DataVariables 20
4.5 TypeDefinitionNodes 20
4.5.1 General 20
4.5.2 Complex TypeDefinitionNodes and their InstanceDeclarations 21
4.5.3 Subtyping 22
4.5.4 Instantiation of complex TypeDefinitionNodes 22
4.6 Event Model 23
4.6.1 General 23
4.6.2 EventTypes 24
4.6.3 Event Categorization 24
4.7 Methods 25
4.8 Roles 25
4.8.1 Overview 25
4.8.2 Well Known Roles 25
4.8.3 Evaluating Permissions with Roles 26
5 Standard NodeClasses 28
5.1 Overview 28
5.2 Base NodeClass 28
5.2.1 General 28
5.2.2 NodeId 28
5.2.3 NodeClass 29
5.2.4 BrowseName 29
5.2.5 DisplayName 29

AVA Documents
Document Sharing Hub

5.2.6 Description 29
5.2.7 WriteMask 29
5.2.8 UserWriteMask 30
5.2.9 RolePermissions 30
5.2.10 UserRolePermissions 31
5.2.11 AccessRestrictions 31
5.3 ReferenceType NodeClass 31
5.3.1 General 31
5.3.2 Attributes 32
5.3.3 References 33
5.4 View NodeClass 34
5.5 Objects 36
5.5.1 Object NodeClass 36
5.5.2 ObjectType NodeClass 38
5.5.3 Standard ObjectType FolderType 39
5.5.4 Client-side creation of Objects of an ObjectType 39
5.6 Variables 40
5.6.1 General 40
5.6.2 Variable NodeClass 40
5.6.3 Properties 43
5.6.4 DataVariable 43
5.6.5 VariableType NodeClass 44
5.6.6 Client-side creation of Variables of an VariableType 46
5.7 Method NodeClass 46
5.8 DataTypes 48
5.8.1 DataType Model 48
5.8.2 Encoding Rules for different kinds of DataTypes 49
5.8.3 DataType NodeClass 50
5.8.4 DataTypeEncoding and Encoding Information 52
5.9 Summary of Attributes of the NodeClasses 53
6 Type Model for ObjectTypes and VariableTypes 53
6.1 Overview 53
6.2 Definitions 53
6.2.1 InstanceDeclaration 53
6.2.2 Instances without ModellingRules 53
6.2.3 InstanceDeclarationHierarchy 54
6.2.4 Similar Node of InstanceDeclaration 54
6.2.5 BrowsePath 54
6.2.6 Attribute Handling of InstanceDeclarations 54
6.2.7 Attribute Handling of Variable and VariableTypes 54
6.2.8 NodeIds of InstanceDeclarations 54
6.3 Subtyping of ObjectTypes and VariableTypes 54
6.3.1 Overview 54
6.3.2 Attributes 55
6.3.3 InstanceDeclarations 55
6.4 Instances of ObjectTypes and VariableTypes 58
6.4.1 Overview 58
6.4.2 Creating an Instance 58
6.4.3 Constraints on an Instance 59

AVA Documents Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

6.4.4 ModellingRules 59
6.5 Changing Type Definitions that are already used 67
7 Standard ReferenceTypes 67
7.1 General 67
7.2 References ReferenceType 68
7.3 HierarchicalReferences ReferenceType 68
7.4 NonHierarchicalReferences ReferenceType 69
7.5 HasChild ReferenceType 69
7.6 Aggregates ReferenceType 69
7.7 HasComponent ReferenceType 69
7.8 HasProperty ReferenceType 69
7.9 HasOrderedComponent ReferenceType 70
7.10 HasSubtype ReferenceType 70
7.11 Organizes ReferenceType 70
7.12 HasModellingRule ReferenceType 70
7.13 HasTypeDefinition ReferenceType 70
7.14 HasEncoding ReferenceType 71
7.15 GeneratesEvent 71
7.16 AlwaysGeneratesEvent 71
7.17 HasEventSource 71
7.18 HasNotifier 72
8 Standard DataTypes 73
8.1 General 73
8.2 NodeId 73
8.2.1 General 73
8.2.2 NamespaceIndex 73
8.2.3 IdentifierType 74
8.2.4 Identifier value 74
8.3 QualifiedName 75
8.4 LocaleId 75
8.5 LocalizedText 75
8.6 Argument 76
8.7 BaseDataType 76
8.8 Boolean 76
8.9 Byte 76
8.10 ByteString 76
8.11 DateTime 76
8.12 Double 76
8.13 Duration 76
8.14 Enumeration 77
8.15 Float 77
8.16 Guid 77
8.17 SByte 77
8.18 IdType 77
8.19 Image 77
8.20 ImageBMP 77
8.21 ImageGIF 77
8.22 ImageJPG 77
8.23 ImagePNG 77

AVA Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

8.24 Integer 77
8.25 Int16 77
8.26 Int32 77
8.27 Int64 78
8.28 TimeZoneDataType 78
8.29 NamingRuleType 78
8.30 NodeClass 78
8.31 Number 78
8.32 String 78
8.33 Structure 78
8.34 UInteger 78
8.35 UInt16 79
8.36 UInt32 79
8.37 UInt64 79
8.38 UtcTime 79
8.39 XmlElement 79
8.40 EnumValueType 79
8.41 OptionSet 80
8.42 Union 80
8.43 DateString 80
8.44 DecimalString 80
8.45 DurationString 80
8.46 NormalizedString 81
8.47 TimeString 81
8.48 DataTypeDefinition 81
8.49 StructureDefinition 81
8.50 EnumDefinition 82
8.51 StructureField 82
8.52 EnumField 83
8.53 AudioDataType 83
8.54 Decimal 83
8.55 PermissionType 83
8.56 AccessRestrictionsType 84
8.57 AccessLevelType 84
8.58 AccessLevelExType 85
8.59 EventNotifierType 85
8.60 AttributeWriteMask 86
9 Standard EventTypes 87
9.1 General 87
9.2 BaseEventType 87
9.3 SystemEventType 87
9.4 ProgressEventType 88
9.5 AuditEventType 88
9.6 AuditSecurityEventType 89
9.7 AuditChannelEventType 89
9.8 AuditOpenSecureChannelEventType 89
9.9 AuditSessionEventType 89
9.10 AuditCreateSessionEventType 90
9.11 AuditUrlMismatchEventType 90

AVA Documents Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

9.12 AuditActivateSessionEventType 90
9.13 AuditCancelEventType 90
9.14 AuditCertificateEventType 90
9.15 AuditCertificateDataMismatchEventType 90
9.16 AuditCertificateExpiredEventType 90
9.17 AuditCertificateInvalidEventType 90
9.18 AuditCertificateUntrustedEventType 90
9.19 AuditCertificateRevokedEventType 90
9.20 AuditCertificateMismatchEventType 90
9.21 AuditNodeManagementEventType 91
9.22 AuditAddNodesEventType 91
9.23 AuditDeleteNodesEventType 91
9.24 AuditAddReferencesEventType 91
9.25 AuditDeleteReferencesEventType 91
9.26 AuditUpdateEventType 91
9.27 AuditWriteUpdateEventType 91
9.28 AuditHistoryUpdateEventType 91
9.29 AuditUpdateMethodEventType 91
9.30 DeviceFailureEventType 91
9.31 SystemStatusChangeEventType 91
9.32 ModelChangeEvents 91
9.32.1 General 91
9.32.2 NodeVersion Property 92
9.32.3 Views 92
9.32.4 Event Compression 92
9.32.5 BaseModelChangeEventType 92
9.32.6 GeneralModelChangeEventType 92
9.32.7 Guidelines for ModelChangeEvents 92
9.33 SemanticChangeEventType 93
9.33.1 General 93
9.33.2 ViewVersion and NodeVersion Properties 93
9.33.3 Views 93
9.33.4 Event Compression 93
Annex A (informative) How to use the Address Space Model 94
A.1 Overview 94
A.2 Type definitions 94
A.3 ObjectTypes 94
A.4 VariableTypes 94
A.4.1 General 94
A.4.2 Properties or DataVariables 94
A.4.3 Many Variables and / or structured DataTypes 95
A.5 Views 95
A.6 Methods 96
A.7 Defining ReferenceTypes 96
A.8 Defining ModellingRules 96
Annex B (informative) OPC UA Meta Model in UML 97
B.1 Background 97
B.2 Notation 97
B.3 Meta Model 98

AVA Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

B.3.1 Base 99
B.3.2 ReferenceType 99
B.3.3 Predefined ReferenceTypes 100
B.3.4 Attributes 100
B.3.5 Object and ObjectType 101
B.3.6 EventNotifier 102
B.3.7 Variable and VariableType 102
B.3.8 Method 103
B.3.9 DataType 104
B.3.10 View 105
Annex C (normative) Graphical Notation 106
C.1 General 106
C.2 Notation 106
C.2.1 Overview 106
C.2.2 Simple Notation 106
C.2.3 Extended Notation 107

Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

FIGURES

Figure 1 – AddressSpace Node diagrams 16
Figure 2 – OPC UA Object Model 18
Figure 3 – AddressSpace Node Model 18
Figure 4 – Reference Model 19
Figure 5 – Example of a Variable defined by a VariableType 21
Figure 6 – Example of a Complex TypeDefinition 21
Figure 7 – Object and its Components defined by an ObjectType 23
Figure 8 – Permissions in the Address Space 31
Figure 9 – Symmetric and Non-Symmetric References 33
Figure 10 – Variables, VariableTypes and their DataTypes 48
Figure 11 – DataType Model 49
Figure 12 – Example of DataType Modelling 52
Figure 13 – Subtyping TypeDefinitionNodes 55
Figure 14 – The Fully-Inherited InstanceDeclarationHierarchy for BetaType 57
Figure 15 – An Instance and its TypeDefinitionNode 58
Figure 16 – Example for several References between InstanceDeclarations 59
Figure 17 – Example on changing instances based on InstanceDeclarations 61
Figure 18 – Example on changing InstanceDeclarations based on an InstanceDeclaration 62
Figure 19 – Use of the Standard ModellingRule Mandatory 63
Figure 20 – Example using the Standard ModellingRules Optional and Mandatory 64
Figure 21 – Example on using ExposesItsArray 65
Figure 22 – Complex example on using ExposesItsArray 65
Figure 23 – Example using OptionalPlaceholder with an Object and Variable 65
Figure 24 – Example using OptionalPlaceholder with a Method 66
Figure 25 – Example on using MandatoryPlaceholder for Object and Variable 67
Figure 26 – Standard ReferenceType Hierarchy 68
Figure 27 – Event Reference Example 72
Figure 28 – Complex Event Reference Example 73
Figure 29 – Standard EventType Hierarchy 87
Figure 30 – Audit Behaviour of a Server 88
Figure 31 – Audit Behaviour of an Aggregating Server 89
Figure B.1 – Background of OPC UA Meta Model 97
Figure B.2 – Notation (I) 98
Figure B.3 – Notation (II) 98
Figure B.4 – Base 99
Figure B.5 – Reference and ReferenceType 99
Figure B.6 – Predefined ReferenceTypes 100
Figure B.7 – Attributes 101
Figure B.8 – Object and ObjectType 102
Figure B.9 – EventNotifier 102
Figure B.10 – Variable and VariableType 103

AVA Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

Figure B.11 – Method 104
Figure B.12 – DataType 105
Figure B.13 – View 105
Figure C.1 – Example of a Reference connecting two Nodes 107
Figure C.2 – Example of using a TypeDefinition inside a Node 108
Figure C.3 – Example of exposing Attributes 108
Figure C.4 – Example of exposing Properties inline 109

Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

TABLES

Table 1 – NodeClass Table Conventions 17
Table 2 – Well-Known Roles 26
Table 3 – Example Roles 26
Table 4 – Example Nodes 27
Table 5 – Example Role Assignment 27
Table 6 – Examples of Evaluating Access 27
Table 7 – Base NodeClass 28
Table 9 – ReferenceType NodeClass 32
Table 10 – View NodeClass 35
Table 11 – Object NodeClass 37
Table 12 – ObjectType NodeClass 38
Table 13 – Variable NodeClass 40
Table 14 – VariableType NodeClass 45
Table 15 – Method NodeClass 47
Table 16 – DataType NodeClass 51
Table 17 – Overview of Attributes 53
Table 18 – The InstanceDeclarationHierarchy for BetaType 56
Table 19 – The Fully-Inherited InstanceDeclarationHierarchy for BetaType 56
Table 20 – Rule for ModellingRules Properties when Subtyping 61
Table 21 – Properties of ModellingRules 62
Table 22 – NodeId Definition 73
Table 23 – IdentifierType Values 74
Table 24 – NodeId Null Values 74
Table 25 – QualifiedName Definition 75
Table 26 – LocaleId Examples 75
Table 27 – LocalizedText Definition 75
Table 28 – Argument Definition 76
Table 29 – TimeZoneDataType Definition 78
Table 30 – NamingRuleType Values 78
Table 31 – NodeClass Values 78
Table 32 – EnumValueType Definition 79
Table 33 – OptionSet Definition 80
Table 34 – StructureDefinition Structure 82
Table 35 – EnumDefinition Structure 82
Table 36 – StructureField Structure 82
Table 37 – EnumField Structure 83
Table 38 – PermissionType Definition 83
Table 39 – AccessRestrictionsType Definition 84
Table 40 – AccessLevelType Definition 85
Table 41 – AccessLevelExType Definition 85
Table 42 – EventNotifierType Definition 86

AVA Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

Table 43 – Bit mask for WriteMask and UserWriteMask 86
Table C.1 – Notation of Nodes depending on the NodeClass 106
Table C.2 – Simple Notation of Nodes depending on the NodeClass 107

Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

OPC UNIFIED ARCHITECTURE –

Part 3: Address Space 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.

The main task of IEC technical committees is to prepare International Standards. However, a technical committee may propose the publication of a technical report when it has collected data of a different kind from that which is normally published as an International Standard, for example "state of the art".

IEC 62541-3 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 technical report is based on the following documents:

Enquiry draft	Report on voting
65E/XX/DTR	65E/XX/RVC

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

This third edition cancels and replaces the second edition of IEC 62541, published in 2015.

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

AVA Documents
Document Sharing Hub

a) Added new improved approach for exposing structure definitions. An Attribute on the DataType Node now simply contains a binary description.

b) Added new flags for Variables to indicate atomicity when reading or writing.

c) Added Roles and Permissions to allow configuration of a role-based authorization.

d) Added new data types: “Union”, “Decimal”, “OptionSet”, “DateString”, “TimeString”, “DurationString”, NormalizedString”, “DecimalString”, and “AudioDataType”.

e) Added definition on how to use the ModellingRules OptionalPlaceHolder and MandatoryPlaceHolder for Methods.

f) Added optional Properties “MaxCharacters” and “MaxByteStringLength” to Variable Nodes.

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

Throughout this document and the referenced other Parts of the series, certain document conventions are used:

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

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

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

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

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

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

The National Committees are requested to note that for this publication the stability date is 2021.

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

A bilingual version of this publication may be issued at a later date.

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.

Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

OPC Unified Architecture Specification

Part 3: Address Space Model

1 Scope

This specification describes the OPC Unified Architecture (OPC UA) *AddressSpace* and its *Objects*. This Part is the OPC UA meta model on which OPC UA information models are based.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application.

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

IEC TR 62541-2, *OPC Unified Architecture – Part 2: Security Model*

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

IEC 62541-5, *OPC Unified Architecture – Part 5: Information Model*

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

IEC 62541-8, *OPC Unified Architecture – Part 8: Data Access*

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

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

ISO/IEC 10918-1: Information technology – Digital compression and coding of continuous-tone still images: Requirements and guidelines
https://www.iso.org/standard/18902.html

ISO/IEC 15948: Information technology – Computer graphics and image processing – Portable Network Graphics (PNG): Functional specification
https://www.iso.org/standard/29581.html

ISO 639 (all parts): Codes for the representation of names of languages
https://www.iso.org/iso-639-language-codes.html

ISO 3166 (all parts): Codes for the representation of names of countries and their subdivisions
https://www.iso.org/iso-3166-country-codes.html

ISO/IEC/IEEE 60559:2011: Information technology – Microprocessor Systems – Floating-Point arithmetic
https://www.iso.org/standard/57469.html

IETF RFC 5646: Tags for Identifying Languages
http://tools.ietf.org/html/rfc5646

ISO 8601-2000: Data elements and interchange formats
https://www.iso.org/standard/26780.html

Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

Unicode Annex15: Unicode Standard Annex #15: Unicode Normalization Forms
http://www.unicode.org/reports/tr15/

W3C XML Schema Definition Language (XSD) Part 2: DataTypes
http://www.w3.org/TR/xmlschema-2/

TAI: International Atomic Time
http://www.bipm.org/en/bipm-services/timescales/tai.html

3 Terms, definitions, abbreviations and conventions

3.1 Terms and definitions

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

3.1.1
DataType
instance of a *DataType Node* that is used together with the *ValueRank Attribute* to define the data type of a *Variable*

3.1.2
DataTypeId
NodeId of a *DataType Node*

3.1.3
DataVariable
Variables that represent *values* of *Objects*, either directly or indirectly for complex *Variables*, where the *Variables* are always the *TargetNode* of a *HasComponent Reference*

3.1.4
EventType
ObjectType Node that represents the type definition of an *Event*

3.1.5
Hierarchical Reference
Reference that is used to construct hierarchies in the *AddressSpace*

Note 1 to entry: All hierarchical ReferenceTypes are derived from HierarchicalReferences.

3.1.6
InstanceDeclaration
Node that is used by a complex *TypeDefinitionNode* to expose its complex structure

Note 1 to entry: It is an instance used by a type definition.

3.1.7
ModellingRule
metadata of an *InstanceDeclaration* that defines how the *InstanceDeclaration* will be used for instantiation and also defines subtyping rules for an *InstanceDeclaration*

3.1.8
Property
Variables that are the *TargetNode* for a *HasProperty Reference*

Note 1 to entry: *Properties* describe the characteristics of a *Node*.

3.1.9
SourceNode
Node having a *Reference* to another *Node*

EXAMPLE: In the *Reference* “A contains B”, “A” is the *SourceNode*.

3.1.10
TargetNode
Node that is referenced by another *Node*

Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

EXAMPLE: In the *Reference* "A Contains B", "B" is the *TargetNode*.

3.1.11
TypeDefinitionNode
Node that is used to define the type of another *Node*

Note 1 to entry: *ObjectType* and *VariableType Nodes* are *TypeDefinitionNodes*.

3.1.12
VariableType
Node that represents the type definition for a *Variable*

3.2 Abbreviations

UA Unified Architecture
UML Unified Modeling Language
URI Uniform Resource Identifier
W3C World Wide Web Consortium
XML Extensible Markup Language

3.3 Conventions

3.3.1 Conventions for AddressSpace figures

Nodes and their *References* to each other are illustrated using figures. Figure 1 illustrates the conventions used in these figures.

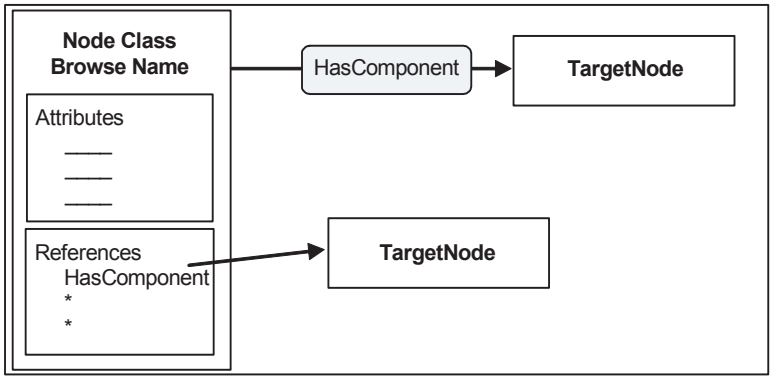

Figure 1 – AddressSpace Node diagrams

In these figures, rectangles represent *Nodes*. *Node* rectangles may be titled with one or two lines of text. When two lines are used, the first text line in the rectangle identifies the *NodeClass* and the second line contains the *BrowseName*. When one line is used, it contains the *BrowseName*.

Node rectangles may contain boxes used to define their *Attributes* and *References*. Specific names in these boxes identify specific *Attributes* and *References*.

Shaded rectangles with rounded corners and with arrows passing through them represent *References*. The arrow that passes through them begins at the *SourceNode* and points to the *TargetNode*. *References* may also be shown by drawing an arrow that starts at the *Reference* name in the "References" box and ends at the *TargetNode*.

3.3.2 Conventions for defining NodeClasses

Clause 5 defines *AddressSpace NodeClasses*. Table 1 describes the format of the tables used to define *NodeClasses*.

Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

Table 1 – NodeClass Table Conventions

Name	Use	Data Type	Description
Attributes			
"Attribute name"	"M" or "O"	Data type of the *Attribute*	Defines the *Attribute*
References			
"Reference name"	"1", "0..1" or "0..*"	Not used	Describes the use of the *Reference* by the *NodeClass*
Standard Properties			
"Property name"	"M" or "O"	Data type of the *Property*	Defines the *Property*

The Name column contains the name of the *Attribute*, the name of the *ReferenceType* used to create a *Reference* or the name of a *Property* referenced using the *HasProperty Reference*.

The Use column defines whether the *Attribute* or *Property* is mandatory (M) or optional (O). When mandatory the *Attribute* or *Property* shall exist for every *Node* of the *NodeClass*. For *References* it specifies the cardinality. The following values may apply:

- "0..*" identifies that there are no restrictions, that is, the *Reference* does not have to be provided but there is no limitation how often it can be provided;
- "0..1" identifies that the *Reference* is provided at most once;
- "1" identifies that the *Reference* shall be provided exactly once.

The Data Type column contains the name of the *DataType* of the *Attribute* or *Property*. It is not used for *References*.

The Description column contains the description of the *Attribute*, the *Reference* or the *Property*.

Only this standard may define *Attributes*. Thus, all *Attributes* of the *NodeClass* are specified in the table and can only be extended by other parts of this series of standards.

This standard also defines *ReferenceTypes*, but *ReferenceTypes* can also be specified by a *Server* or by a client using the *NodeManagement Services* specified in IEC 62541-4. Thus, the *NodeClass* tables contained in this standard can contain the base *ReferenceType* called *References* identifying that any *ReferenceType* may be used for the *NodeClass,* including system specific *ReferenceTypes*. The *NodeClass* tables only specify how the *NodeClasses* can be used as *SourceNodes* of *References*, not as *TargetNodes*. If a *NodeClass* table allows a *ReferenceType* for its *NodeClass* to be used as *SourceNode*, this is also true for subtypes of the *ReferenceType*. However, subtypes of the *ReferenceType* may restrict its *SourceNodes*.

This standard defines *Properties*, but *Properties* can be defined by other standard organizations or vendors and *Nodes* can have *Properties* that are not standardised. *Properties* defined in this standard are defined by their name, which is mapped to the *BrowseName* having the *NamespaceIndex* 0, which represents the *Namespace* for OPC UA.

The Use column (optional or mandatory) does not imply a specific *ModellingRule* for *Properties*. Different *Server* implementations will choose to use *ModellingRules* appropriate for them.

4 AddressSpace concepts

4.1 Overview

The remainder of 4 defines the concepts of the *AddressSpace*. Clause 5 defines the *NodeClasses* of the *AddressSpace* representing the *AddressSpace* concepts. Clause 6 defines details on the type model for *ObjectTypes* and *VariableTypes*. Standard *ReferenceTypes*, *DataTypes* and *EventTypes* are defined in Clauses 7 to 9.

Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

The informative Annex A describes general considerations on how to use the Address Space Model and the informative Annex B provides a UML Model of the Address Space Model. The normative Annex C defines a graphical notation for OPC UA data.

4.2 Object Model

The primary objective of the OPC UA *AddressSpace* is to provide a standard way for *Servers* to represent *Objects* to *Clients*. The OPC UA Object Model has been designed to meet this objective. It defines *Objects* in terms of *Variables* and *Methods*. It also allows relationships to other *Objects* to be expressed. Figure 2 illustrates the model.

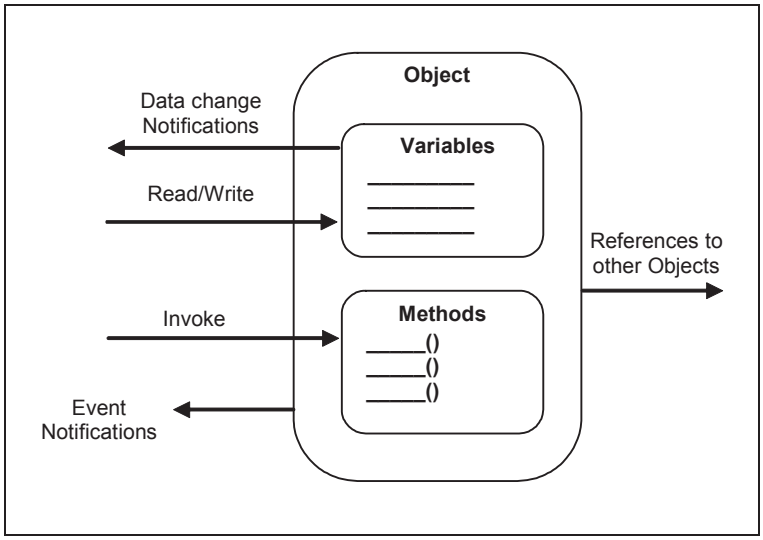

Figure 2 – OPC UA Object Model

The elements of this model are represented in the *AddressSpace* as *Nodes*. Each *Node* is assigned to a *NodeClass* and each *NodeClass* represents a different element of the Object Model. Clause 5 defines the *NodeClasses* used to represent this model.

4.3 Node Model

4.3.1 General

The set of *Objects* and related information that the OPC UA *Server* makes available to *Clients* is referred to as its *AddressSpace*. The model for *Objects* is defined by the OPC UA Object Model (see 4.2).

Objects and their components are represented in the *AddressSpace* as a set of *Nodes* described by *Attributes* and interconnected by *References*. Figure 3 illustrates the model of a *Node* and the remainder of 4.3 discusses the details of the Node Model.

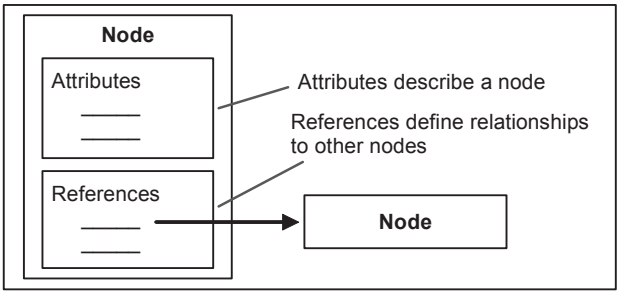

Figure 3 – AddressSpace Node Model

4.3.2 NodeClasses

NodeClasses are defined in terms of the *Attributes* and *References* that shall be instantiated (given values) when a *Node* is defined in the *AddressSpace*. *Attributes* are discussed in 4.3.3 and *References* in 4.3.4.

AVA Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

Clause 5 defines the *NodeClasses* for the OPC UA *AddressSpace*. These *NodeClasses* are referred to collectively as the metadata for the *AddressSpace*. Each *Node* in the *AddressSpace* is an instance of one of these *NodeClasses*. No other *NodeClasses* shall be used to define *Nodes*, and as a result, *Clients* and *Servers* are not allowed to define *NodeClasses* or extend the definitions of these *NodeClasses*.

4.3.3 Attributes

Attributes are data elements that describe *Nodes*. *Clients* can access *Attribute* values using Read, Write, Query, and Subscription/MonitoredItem *Services*. These *Services* are defined in IEC 62541-4.

Attributes are elementary components of *NodeClasses*. *Attribute* definitions are included as part of the *NodeClass* definitions in Clause 5 and, therefore, are not included in the *AddressSpace*.

Each *Attribute* definition consists of an attribute id (for attribute ids of *Attributes*, see IEC 62541-6), a name, a description, a data type and a mandatory/optional indicator. The set of *Attributes* defined for each *NodeClass* shall not be extended by *Clients* or *Servers*.

When a *Node* is instantiated in the *AddressSpace*, the values of the *NodeClass Attributes* are provided. The mandatory/optional indicator for the *Attribute* indicates whether the *Attribute* has to be instantiated.

4.3.4 References

References are used to relate *Nodes* to each other. They can be accessed using the browsing and querying *Services* defined in IEC 62541-4.

Like *Attributes*, they are defined as fundamental components of *Nodes*. Unlike *Attributes*, *References* are defined as instances of *ReferenceType Nodes*. *ReferenceType Nodes* are visible in the *AddressSpace* and are defined using the *ReferenceType NodeClass* (see 0).

The *Node* that contains the *Reference* is referred to as the *SourceNode* and the *Node* that is referenced is referred to as the *TargetNode*. The combination of the *SourceNode*, the *ReferenceType* and the *TargetNode* are used in OPC UA *Services* to uniquely identify *References*. Thus, each *Node* can reference another *Node* with the same *ReferenceType* only once. Any subtypes of concrete *ReferenceTypes* are considered to be equal to the base concrete *ReferenceTypes* when identifying *References* (see 0 for subtypes of *ReferenceTypes*). Figure 4 illustrates this model of a *Reference*.

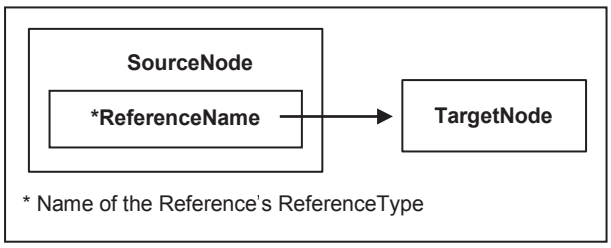

Figure 4 – Reference Model

The *TargetNode* of a *Reference* may be in the same *AddressSpace* or in the *AddressSpace* of another OPC UA *Server*. *TargetNodes* located in other *Servers* are identified in OPC UA *Services* using a combination of the remote *Server* name and the identifier assigned to the *Node* by the remote *Server*.

OPC UA does not require that *the TargetNode* exists, thus *References* may point to a *Node* that does not exist.

AVA Documents Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

4.4 Variables

4.4.1 General

Variables are used to represent *values*. Two types of *Variables* are defined, *Properties* and *DataVariables*. They differ in the kind of data that they represent and whether they can contain other *Variables*.

4.4.2 Properties

Properties are *Server*-defined characteristics of *Objects*, *DataVariables* and other *Nodes*. *Properties* differ from *Attributes* in that they characterise *what* the *Node* represents, such as a device or a purchase order. *Attributes* define additional metadata that is instantiated for all *Nodes* from a *NodeClass*. *Attributes* are common to all *Nodes* of a *NodeClass* and only defined by this specification whereas *Properties* can be *Server*-defined.

For example, an *Attribute* defines the *DataType* of *Variables* whereas a *Property* can be used to specify the engineering unit of some *Variables*.

To prevent recursion, *Properties* are not allowed to have *Properties* defined for them. To easily identify *Properties*, the *BrowseName* of a *Property* shall be unique in the context of the *Node* containing the *Properties* (see 0 for details).

A *Node* and its *Properties* shall always reside in the same *Server*.

4.4.3 DataVariables

DataVariables represent the content of an *Object*. For example, a file *Object* may be defined that contains a stream of bytes. The stream of bytes may be defined as a *DataVariable* that is an array of bytes. *Properties* may be used to expose the creation time and owner of the file *Object*.

For example, if a *DataVariable* is defined by a data structure that contains two fields, “startTime” and “endTime” then it might have a *Property* specific to that data structure, such as “earliestStartTime”.

As another example, function blocks in control systems might be represented as *Objects*. The parameters of the function block, such as its setpoints, may be represented as *DataVariables*. The function block *Object* might also have *Properties* that describe its execution time and its type.

DataVariables may have additional *DataVariables*, but only if they are complex. In this case, their *DataVariables* shall always be elements of their complex definitions. Following the example introduced by the description of *Properties* in 4.4.2, the *Server* could expose “startTime” and “endTime” as separate components of the data structure.

As another example, a complex *DataVariable* may define an aggregate of temperature values generated by three separate temperature transmitters that are also visible in the *AddressSpace*. In this case, this complex *DataVariable* could define *HasComponent References* from it to the individual temperature values that it is composed of.

4.5 TypeDefinitionNodes

4.5.1 General

OPC UA *Servers* shall provide type definitions for *Objects* and *Variables*. The *HasTypeDefinition Reference* shall be used to link an instance with its type definition represented by a *TypeDefinitionNode*. Type definitions are required; however, IEC 62541-5 defines a *BaseObjectType*, a *PropertyType,* and a *BaseDataVariableType* so a *Server* can use such a base type if no more specialised type information is available. *Objects* and *Variables* inherit the *Attributes* specified by their *TypeDefinitionNode* (see 6.4 for details).

In some cases, the *NodeId* used by the *HasTypeDefinition Reference* will be well-known to *Clients* and *Servers*. Organizations may define *TypeDefinitionNodes* that are well-known in the industry. Well-known *NodeIds* of *TypeDefinitionNodes* provide for commonality across OPC UA *Servers* and allow *Clients* to interpret the *TypeDefinitionNode* without having to read

AVA Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

it from the *Server*. Therefore, *Servers* may use well-known *NodeIds* without representing the corresponding *TypeDefinitionNodes* in their *AddressSpace*. However, the *TypeDefinitionNodes* shall be provided for generic *Clients*. These *TypeDefinitionNodes* may exist in another *Server*.

The following example, illustrated in Figure 5, describes the use of the *HasTypeDefinition Reference*. In this example, a setpoint parameter "SP" is represented as a *DataVariable* in the *AddressSpace*. This *DataVariable* is part of an *Object* not shown in the figure.

To provide for a common setpoint definition that can be used by other *Objects*, a specialised *VariableType* is used. Each setpoint *DataVariable* that uses this common definition will have a *HasTypeDefinition Reference* that identifies the common "SetPoint" *VariableType*.

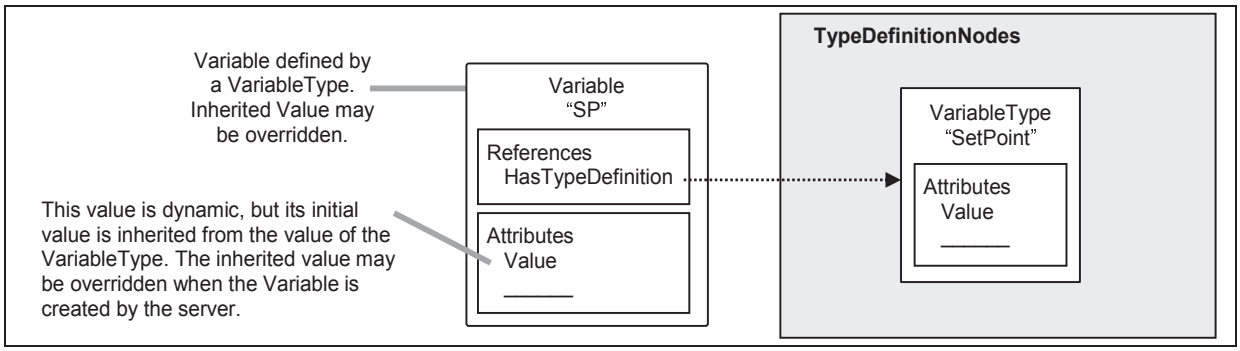

Figure 5 – Example of a Variable defined by a VariableType

4.5.2 Complex TypeDefinitionNodes and their InstanceDeclarations

TypeDefinitionNodes can be complex. A complex *TypeDefinitionNode* also defines *References* to other *Nodes* as part of the type definition. The *ModellingRules* defined in 6.4.4 specify how those *Nodes* are handled when creating an instance of the type definition.

A *TypeDefinitionNode* references instances instead of other *TypeDefinitionNodes* to allow unique names for several instances of the same type, to define default values and to add *References* for those instances that are specific to this complex *TypeDefinitionNode* and not to the *TypeDefinitionNode* of the instance. For example, in Figure 6 the *ObjectType* "AI_BLK_TYPE", representing a function block, has a *HasComponent Reference* to a *Variable* "SP" of the *VariableType* "SetPoint". "AI_BLK_TYPE" could have an additional setpoint *Variable* of the same type using a different name. It could add a *Property* to the *Variable* that was not defined by its *TypeDefinitionNode* "SetPoint". And it could define a default value for "SP", that is, each instance of "AI_BLK_TYPE" would have a *Variable* "SP" initially set to this value.

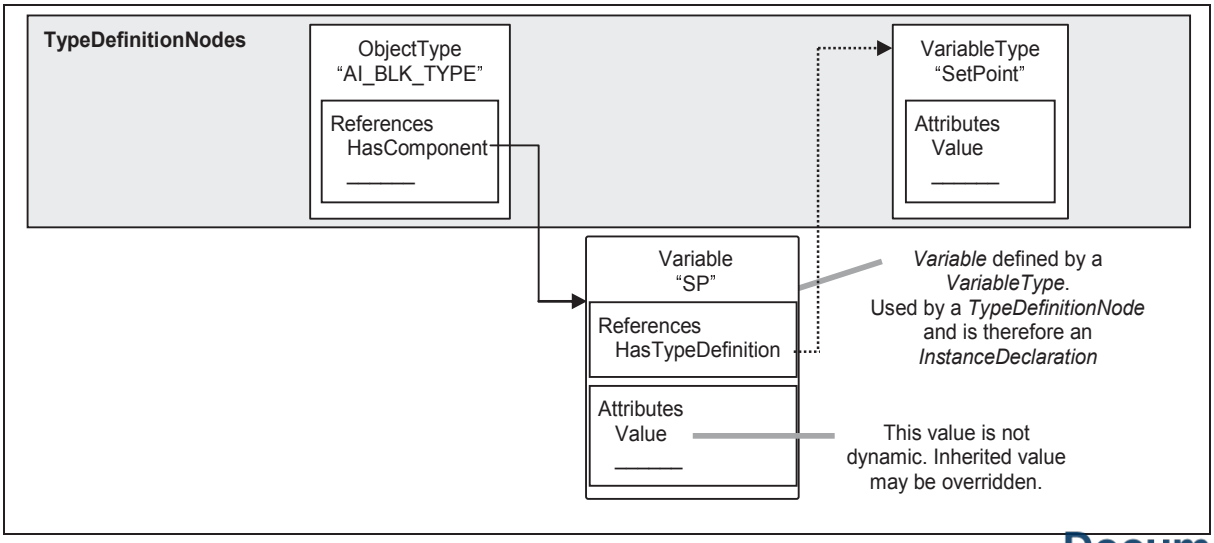

Figure 6 – Example of a Complex TypeDefinition

Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

This approach is commonly used in object-oriented programming languages in which the variables of a class are defined as instances of other classes. When the class is instantiated, each variable is also instantiated, but with the default values (constructor values) defined for the containing class. That is, typically, the constructor for the component class runs first, followed by the constructor for the containing class. The constructor for the containing class may override component values set by the component class.

To distinguish instances used for the type definitions from instances that represent real data, those instances are called *InstanceDeclarations*. However, this term is used to simplify this specification, if an instance is an *InstanceDeclaration* or not is only visible in the *AddressSpace* by following its *References*. Some instances may be shared and therefore referenced by *TypeDefinitionNodes*, *InstanceDeclarations* and instances. This is similar to class variables in object-oriented programming languages.

4.5.3 Subtyping

This standard allows subtyping of type definitions. The subtyping rules are defined in Clause 6. Subtyping of *ObjectTypes* and *VariableTypes* allows:

- *Clients* that only know the supertype to handle an instance of the subtype as if it were an instance of the supertype;
- instances of the supertype to be replaced by instances of the subtype;
- specialised types that inherit common characteristics of the base type.

In other words, subtypes reflect the structure defined by their supertype but may add additional characteristics. For example, a vendor may wish to extend a general “TemperatureSensor” *VariableType* by adding a *Property* providing the next maintenance interval. The vendor would do this by creating a new *VariableType* which is a *TargetNode* for a *HasSubtype* reference from the original *VariableType* and adding the new *Property* to it.

4.5.4 Instantiation of complex TypeDefinitionNodes

The instantiation of complex *TypeDefinitionNodes* depends on the *ModellingRules* defined in 6.4.4. However, the intention is that instances of a type definition will reflect the structure defined by the *TypeDefinitionNode*. Figure 7 shows an instance of the *TypeDefinitionNode* “AI_BLK_TYPE”, where the *ModellingRule Mandatory*, defined in 6.4.4.5.2, was applied for its containing *Variable*. Thus, an instance of “AI_BLK_TYPE”, called AI_BLK_1”, has a *HasTypeDefinition Reference* to “AI_BLK_TYPE”. It also contains a *Variable* “SP” having the same *BrowseName* as the *Variable* “SP” used by the *TypeDefinitionNode* and thereby reflects the structure defined by the *TypeDefinitionNode*.

Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

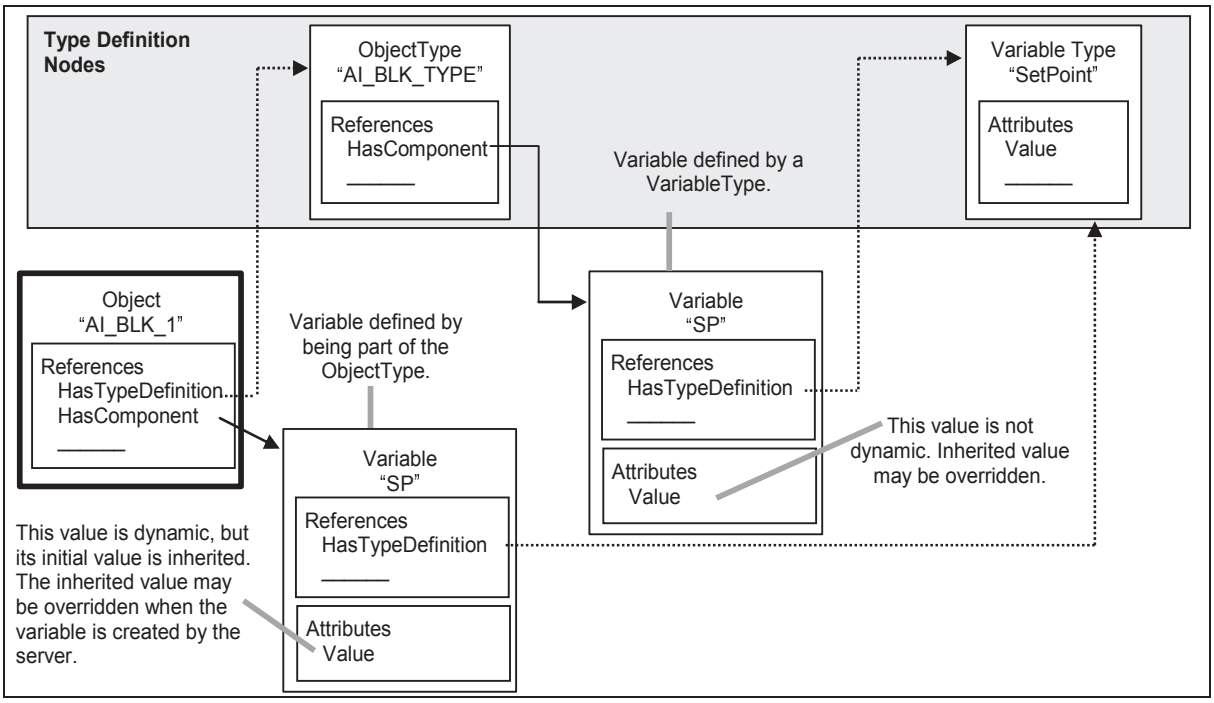

Figure 7 – Object and its Components defined by an ObjectType

A client knowing the *ObjectType* "AI_BLK_TYPE" can use this knowledge to directly browse to the containing *Nodes* for each instance of this type. This allows programming against the *TypeDefinitionNode*. For example, a graphical element may be programmed in the client that handles all instances of "AI_BLK_TYPE" in the same way by showing the value of "SP".

There are several constraints related to programming against the *TypeDefinitionNode*. A *TypeDefinitionNode* or an *InstanceDeclaration* shall never reference two *Nodes* having the same *BrowseName* using forward *hierarchical References*. Instances based on *InstanceDeclarations* shall always keep the same *BrowseName* as the *InstanceDeclaration* they are derived from. A special *Service* defined in IEC 62541-4 called TranslateBrowsePathsToNodeIds may be used to identify the instances based on the *InstanceDeclarations*. Using the simple Browse *Service* might not be sufficient since the uniqueness of the *BrowseName* is only required for *TypeDefinitionNodes* and *InstanceDeclarations*, not for other instances. Thus, "AI_BLK_1" may have another *Variable* with the *BrowseName* "SP", although this one would not be derived from an *InstanceDeclaration* of the *TypeDefinitionNode*.

Instances derived from an *InstanceDeclaration* shall be of the same *TypeDefinitionNode* or a subtype of this *TypeDefinitionNode*.

A *TypeDefinitionNode* and its *InstanceDeclarations* shall always reside in the same *Server*. However, instances may point with their *HasTypeDefinition Reference* to a *TypeDefinitionNode* in a different *Server*.

4.6 Event Model

4.6.1 General

The Event Model defines a general purpose eventing system that can be used in many diverse vertical markets.

Events represent specific transient occurrences. System configuration changes and system errors are examples of *Events*. *Event Notifications* report the occurrence of an *Event*. *Events* defined in this document are not directly visible in the OPC UA *AddressSpace*. *Objects* and *Views* can be used to subscribe to *Events*. The *EventNotifier Attribute* of those *Nodes* identifies if the *Node* allows subscribing to *Events*. *Clients* subscribe to such *Nodes* to receive *Notifications* of *Event* occurrences.

Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

Event Subscriptions use the Monitoring and Subscription *Services* defined in IEC 62541-4 to subscribe to the *Event Notifications* of a *Node*.

Any OPC UA *Server* that supports eventing shall expose at least one *Node* as *EventNotifier*. The *Server Object* defined in IEC 62541-5 is used for this purpose. *Events* generated by the *Server* are available via this *Server Object*. A *Server* is not expected to produce *Events* if the connection to the event source is down for some reason (i.e. the system is offline).

Events may also be exposed through other *Nodes* anywhere in the *AddressSpace*. These *Nodes* (identified via the *EventNotifier Attribute*) provide some subset of the *Events* generated by the *Server*. The position in the *AddressSpace* dictates what this subset will be. For example, a process area *Object* representing a functional area of the process would provide *Events* originating from that area of the process only. It should be noted that this is only an example and it is fully up to the *Server* to determine what *Events* should be provided by which *Node*.

4.6.2 EventTypes

Each *Event* is of a specific *EventType*. A *Server* may support many types. This part defines the *BaseEventType* that all other *EventTypes* derive from. It is expected that other companion specifications will define additional *EventTypes* deriving from the base types defined in this part.

The *EventTypes* supported by a *Server* are exposed in the *AddressSpace* of a *Server*. *EventTypes* are represented as *ObjectTypes* in the *AddressSpace* and do not have a special *NodeClass* associated to them. IEC 62541-5 defines how a *Server* exposes the *EventTypes* in detail.

EventTypes defined in this document are specified as abstract and therefore never instantiated in the *AddressSpace*. Event occurrences of those *EventTypes* are only exposed via a *Subscription*. *EventTypes* exist in the *AddressSpace* to allow *Clients* to discover the *EventType*. This information is used by a client when establishing and working with *Event Subscriptions*. *EventTypes* defined by other parts of this series of standards or companion specifications as well as *Server* specific *EventTypes* may be defined as not abstract and therefore instances of those *EventTypes* may be visible in the *AddressSpace* although *Events* of those *EventTypes* are also accessible via the *Event Notification* mechanisms.

Standard *EventTypes* are described in Clause 9. Their representation in the *AddressSpace* is specified in IEC 62541-5.

4.6.3 Event Categorization

Events can be categorised by creating new *EventTypes* which are subtypes of existing *EventTypes* but do not extend an existing type. They are used only to identify an event as being of the new *EventType*. For example, the *EventType* DeviceFailureEventType could be subtyped into TransmitterFailureEventType and ComputerFailureEventType. These new subtypes would not add new *Properties* or change the semantic inherited from the DeviceFailureEventType other than purely for categorization of the *Events*.

Event sources can also be organised into groups by using the *Event ReferenceTypes* described in 7.16 and 7.18. For example, a *Server* may define *Objects* in the *AddressSpace* representing *Events* related to physical devices, or *Event* areas of a plant or functionality contained in the *Server*. *Event References* would be used to indicate which *Event* sources represent physical devices and which ones represent some *Server*-based functionality. In addition, *References* can be used to group the physical devices or *Server*-based functionality into hierarchical *Event* areas. In some cases, an *Event* source may be categorised as being both a device and a *Server* function. In this case, two relationships would be established. Refer to the description of the *Event ReferenceTypes* for additional examples.

Clients can select a category or categories of *Events* by defining content filters that include terms specifying the *EventType* of the *Event* or a grouping of *Event* sources. The two mechanisms allow for a single *Event* to be categorised in multiple manners. A client could obtain all *Events* related to a physical device or all failures of a particular device.

Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

4.7 Methods

Methods are “lightweight” functions, whose scope is bounded by an owning (see Note) *Object*, similar to the methods of a class in object-oriented programming or an owning *ObjectType*, similar to static methods of a class. *Methods* are invoked by a client, proceed to completion on the *Server* and return the result to the client. The lifetime of the *Method's* invocation instance begins when the client calls the *Method* and ends when the result is returned.

NOTE The owning *Object* or *ObjectType* is specified in the service call when invoking the *Method*.

While *Methods* may affect the state of the owning *Object*, they have no explicit state of their own. In this sense, they are stateless. *Methods* can have a varying number of input arguments and return resultant arguments. Each *Method* is described by a *Node* of the *Method NodeClass*. This *Node* contains the metadata that identifies the *Method's* arguments and describes its behaviour.

Methods are invoked by using the Call *Service* defined in IEC 62541-4.

Clients discover the *Methods* supported by a *Server* by browsing for the owning *Objects References* that identify their supported *Methods*.

4.8 Roles

4.8.1 Overview

A *Role* is a function assumed by a *Client* when it accesses a *Server*. *Roles* are used to separate authentication (determining who a *Client* is) from authorization (determining what the *Client* is allowed to do). By separating these tasks *Servers* can allow centralized services to manage user identities and credentials while the *Server* only manages the *Permissions* on its *Nodes* assigned to *Roles*.

The set of *Roles* supported by a *Server* are published as components of the *Roles Object* defined in IEC 62541-5. *Servers* should define a base set of *Roles* and allow configuration *Clients* to add system specific *Roles*.

When a *Session* is created, the *Server* must determine what *Roles* are granted to that *Session*. This specification defines standard mapping rules which *Servers* may support. *Servers* may also use vendor specific mapping rules in addition to or instead of the standard rules.

The standard mapping rules allow *Roles* to be granted based on:

- User identity;
- Application identity;
- Endpoint;

User identity mappings can be based on user names, user certificates or user groups. Well known groups include ‘AuthenticatedUser’ (any user with valid credentials) and ‘Anonymous’ (no user credentials provided).

Application identity mappings are based on the *ApplicationUri* specified in the *Client Certificate*. Application identity can only be enforced if the *Client* proves possession of a trusted *Certificate* by using it to create a *Secure Channel* or by providing a signature in *ActivateSession* (see IEC 62541-4).

Endpoint identity mappings are based on the URL used to connect to the *Server*. Endpoint identity can be used to restrict access to *Clients* running on particular networks.

IEC 62541-5 defines the *Objects*, *Methods* and *DataTypes* used to represent and manage these mapping rules in the *Address Space*.

4.8.2 Well Known Roles

All *Servers* should support the well-known *Roles* which are defined in Table 2. The *NodeIds* for the well-known *Roles* are defined in IEC 62541-6.

AVA Documents Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

557 **Table 2 – Well-Known Roles**

BrowseName	Suggested Permissions
Anonymous	The *Role* has very limited access for use when a *Session* has anonymous credentials.
AuthenticatedUser	The *Role* has limited access for use when a *Session* has valid non-anonymous credentials but has not been explicitly granted access to a *Role*.
Observer	The *Role* is allowed to browse, read live data, read historical data/events or subscribe to data/events.
Operator	The *Role* is allowed to browse, read live data, read historical data/events or subscribe to data/events. In addition, the *Session* is allowed to write some live data and call some *Methods*.
Engineer	The *Role* is allowed to browse, read/write configuration data, read historical data/events, call *Methods* or subscribe to data/events.
Supervisor	The *Role* is allowed to browse, read live data, read historical data/events, call *Methods* or subscribe to data/events.
ConfigureAdmin	The *Role* is allowed to change the non-security related configuration settings.
SecurityAdmin	The *Role* is allowed to change security related settings.

558

559 ### 4.8.3 Evaluating Permissions with Roles

560 When a *Client* attempts to access a *Node,* the *Server* goes through the list of *Roles* granted to
561 the *Session* and logically ORs the *Permissions* for the *Role* on the *Node. If there are no Node*
562 *specific Permissions then* the default *Permissions* for the *Role* in the *DefaultRolePermissions*
563 *Property* of the *NamespaceMetadata* for the namespace the *Node* belongs to are used (see
564 IEC 62541-5). The resulting mask is the effective *Permissions*. If the bits corresponding to
565 current operation are set, then the operation can proceed. If they are not set the *Server*
566 returns *Bad_UserAccessDenied*.

567 *Roles* appear under the *Roles Object* in the *Server Address Space*. Each *Role* has mapping
568 rules defined which appear as *Properties* of the *Role Object* (see IEC 62541-5). The examples
569 shown in Table 3 illustrate how the standard mapping rules can be used to determine which
570 *Roles* a *Session* has access to and, consequently, the *Permissions* that are granted to the
571 *Session*.

572 **Table 3 – Example Roles**

Role	Mapping Rules	Description
Anonymous	Identities = Anonymous Applications = Endpoints =	An identity mapping rule that specifies the *Role* applies to anonymous users.
AuthenticatedUser	Identities = AuthenticatedUser Applications = Endpoints =	An identity mapping rule that specifies the *Role* applies to authenticated users.
Operator1	Identities = User with name 'Joe' Applications = urn:OperatorStation1 Endpoints =	An identity mapping rule that specifies specific users that have access to the *Role* with a application rule that restricts access to a single Client application.
Operator2	Identities = Users with name 'Joe' or 'Ann' Applications = urn:OperatorStation2 Endpoints =	An identity mapping rule that specifies specific users that have access to the Role with a application rule that restricts access to a single Client application.
Supervisor	Identities = User with name 'Root' Applications = Endpoints =	An identity mapping rule that specifies specific users that have access to the Role
Administrator	Identities = User with name 'Root'	An identity mapping rule that

AVA Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

	Applications = Endpoints = opc.tcp://127.0.0.1:48000	specifies specific users that have access to the Role when they connect via a specific Endpoint.

The examples also make use of the *Nodes* defined in Table 4. The table specifies the value of the *RolePermissions Attribute* for each *Node*.

Table 4 – Example Nodes

Node	Role Permissions
Unit1.Measurement	AuthenticatedUser = Browse Operator1 = Browse, Read
Unit2.Measurement	AuthenticatedUser = Browse Operator2 = Browse, Read
SetPoint	AuthenticatedUser = Browse Operator1 and Operator2 = Browse, Read, Write Supervisor = Browse, Read
DisableDevice	AuthenticatedUser = Browse Operator1 and Operator2 = Browse, Read Administrator = Browse, Read, Write

When a *Client* creates a *Session* the *Roles* assigned to the *Session* depend on the rules defined for each *Role*. Table 5 lists the assigned *Roles* for different *Sessions* created with different *Users*, *Client* applications and *Endpoints*.

Table 5 – Example Role Assignment

User Provided by Client	Roles Assigned to Session
Anonymous	Anonymous
Sam	AuthenticatedUser
Joe using OperatorStation1 application.	AuthenticatedUser, Operator1
Joe using OperatorStation2 application.	AuthenticatedUser, Operator2
Joe using generic application.	AuthenticatedUser
Root using OperatorStation1 application.	AuthenticatedUser, Supervisor
Root using generic application and 127.0.0.1 endpoint.	AuthenticatedUser, Supervisor, Administrator
Root using generic application and another endpoint.	AuthenticatedUser, Supervisor

When a *Client* application accesses a *Node* the *RolePermissions* for the *Node* are compared to the *Roles* assigned to the *Session*. Any *Permissions* available to at least one *Role* is granted to the *Client*. Table 6 provides a number of scenarios and examples and the resulting decision on access.

Table 6 – Examples of Evaluating Access

Use Case	Role Permissions
Anonymous user on localhost browses Unit1.Measurement Node.	Access denied because no rule defined for Anonymous users.
User 'Sam' using OperatorStation1 application browses Unit1.Measurement *Node*.	Allowed because AuthenticatedUser is granted Browse Permission.
User 'Sam' using OperatorStation2 application reads *Value* of Unit1.Measurement *Node*.	Access denied because AuthenticatedUser is not granted Read *Permission*.
User 'Joe' using OperatorStation1 application reads *Value* of	Allowed because Operator1 is granted Read *Permission*.

AVA Documents Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

Unit1.Measurement *Node*.	
User 'Joe' using OperatorStation2 application reads *Value* of Unit1.Measurement *Node*.	Access denied because AuthenticatedUser and Operator2 are not granted Read *Permission*.
User 'Joe' using generic OPC UA application reads *Value* of Measurement *Node*.	Access denied because AuthenticatedUser is not granted Read *Permission*.
User 'Joe' using OperatorStation1 application write *Value* of SetPoint *Node*.	Allowed because Operator1 is granted Write *Permission*.
User 'Root' using OperatorStation1 application write the *Value* of SetPoint *Node*.	Denied because AuthenticatedUser and Supervisor are not granted Write *Permission*.
User 'Joe' using OperatorStation1 application write Value of DisableDevice Node.	Access denied because AuthenticatedUser and Operator1 are not granted Write *Permission*.
User 'Root' using OperatorStation1 application write the *Value* of DisableDevice *Node*.	Access denied because AuthenticatedUser and Supervisor are not granted Write *Permission*.
User 'Root' using endpoint 127.0.0.1 to write *Value* of DisableDevice *Node*.	Allowed because Administrator is granted Write *Permission*.

5 Standard NodeClasses

5.1 Overview

Clause 5 defines the *NodeClasses* used to define *Nodes* in the OPC UA *AddressSpace*. *NodeClasses* are derived from a common *Base NodeClass*. This *NodeClass* is defined first, followed by those used to organise the *AddressSpace* and then by the *NodeClasses* used to represent *Objects*.

The *NodeClasses* defined to represent *Objects* fall into three categories: those used to define instances, those used to define types for those instances and those used to define data types. Subclause 6.3 describes the rules for subtyping and 6.4 the rules for instantiation of the type definitions.

5.2 Base NodeClass

5.2.1 General

The OPC UA Address Space Model defines a *Base NodeClass* from which all other *NodeClasses* are derived. The derived *NodeClasses* represent the various components of the OPC UA Object Model (see 4.2). The *Attributes* of the *Base NodeClass* are specified in Table 7. There are no *References* specified for the *Base NodeClass*.

Table 7 – Base NodeClass

Name	Use	Data Type	Description
Attributes			
NodeId	M	NodeId	See 5.2.2
NodeClass	M	NodeClass	See 5.2.3
BrowseName	M	QualifiedName	See 5.2.4
DisplayName	M	LocalizedText	See 5.2.5
Description	O	LocalizedText	See 5.2.6
WriteMask	O	AttributeWriteMask	See 5.2.7
UserWriteMask	O	AttributeWriteMask	See 0
RolePermissions	O	RolePermissionType[]	See 5.2.9
UserRolePermissions	O	RolePermissionType[]	See 5.2.10
AccessRestrictions	O	AccessRestrictionsType	See 5.2.11
References			No *References* specified for this *NodeClass*

5.2.2 NodeId

Nodes are unambiguously identified using a constructed identifier called the *NodeId*. Some *Servers* may accept alternative *NodeIds* in addition to the canonical *NodeId* represented in

AVA Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

this *Attribute*. A *Server* shall persist the *NodeId* of a *Node*, that is, it shall not generate new *NodeIds* when rebooting. The structure of the *NodeId* is defined in 8.2.

5.2.3 NodeClass

The *NodeClass Attribute* identifies the *NodeClass* of a *Node*. Its data type is defined in 8.30.

5.2.4 BrowseName

Nodes have a *BrowseName Attribute* that is used as a non-localised human-readable name when browsing the *AddressSpace* to create paths out of *BrowseNames*. The TranslateBrowsePathsToNodeIds *Service* defined in IEC 62541-4 can be used to follow a path constructed of *BrowseNames*.

A *BrowseName* should never be used to display the name of a *Node*. The *DisplayName* should be used instead for this purpose.

Unlike *NodeIds*, the *BrowseName* cannot be used to unambiguously identify a *Node*. Different *Nodes* may have the same *BrowseName*.

Subclause 8.3 defines the structure of the *BrowseName*. It contains a namespace and a string. The namespace is provided to make the *BrowseName* unique in some cases in the context of a *Node* (e.g. *Properties* of a *Node*) although not unique in the context of the *Server*. If different organizations define *BrowseNames* for *Properties*, the namespace of the *BrowseName* provided by the organization makes the *BrowseName* unique, although different organizations may use the same string having a slightly different meaning.

Servers may often choose to use the same namespace for the *NodeId* and the *BrowseName*. However, if they want to provide a standard *Property*, its *BrowseName* shall have the namespace of the standards body although the namespace of the *NodeId* reflects something else, for example the local *Server*.

It is recommended that standard bodies defining standard type definitions use their namespace for the *NodeId* of the *TypeDefinitionNode* as well as for the *BrowseName* of the *TypeDefinitionNode*.

The string-part of the *BrowseName* is case sensitive. That is, *Clients* shall consider them case sensitive. *Servers* are allowed to handle *BrowseNames* passed in *Service* requests as case insensitive. Examples are the *TranslateBrowsePathsToNodeIds Service* or *Event* filter.

5.2.5 DisplayName

The *DisplayName Attribute* contains the localised name of the *Node*. *Clients* should use this *Attribute* if they want to display the name of the *Node* to the user. They should not use the *BrowseName* for this purpose. The *Server* may maintain one or more localised representations for each *DisplayName*. *Clients* negotiate the locale to be returned when they open a session with the *Server*. Refer to IEC 62541-4 for a description of session establishment and locales. Subclause 8.5 defines the structure of the *DisplayName*. The string part of the *DisplayName* is restricted to 512 characters.

5.2.6 Description

The optional *Description Attribute* shall explain the meaning of the *Node* in a localised text using the same mechanisms for localisation as described for the *DisplayName* in 5.2.5.

5.2.7 WriteMask

The optional *WriteMask Attribute* exposes the possibilities of a client to write the *Attributes* of the *Node*. The *WriteMask Attribute* does not take any user access rights into account, that is, although an *Attribute* is writable this may be restricted to a certain user/user group.

If the OPC UA *Server* does not have the ability to get the *WriteMask* information for a specific *Attribute* from the underlying system, it should state that it is writable. If a write operation is called on the *Attribute*, the *Server* should transfer this request and return the corresponding *StatusCode* if such a request is rejected. *StatusCodes* are defined in IEC 62541-4.

AVA Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09

The *AttributeWriteMask DataType* is defined in 8.60.

5.2.8 UserWriteMask

The optional *UserWriteMask Attribute* exposes the possibilities of a client to write the *Attributes* of the *Node* taking user access rights into account. It uses the *AttributeWriteMask DataType* which is defined in 8.60.

The *UserWriteMask Attribute* can only further restrict the *WriteMask Attribute,* when it is set to not writable in the general case that applies for every user.

Clients cannot assume an *Attribute* can be written based on the *UserWriteMask Attribute*.It is possible that the *Server* may return an access denied error due to some server specific change which was not reflected in the state of this *Attribute* at the time the *Client* accessed it.

5.2.9 RolePermissions

The optional *RolePermissions Attribute* specifies the *Permissions* that apply to a *Node* for all *Roles* which have access to the *Node*. The value of the *Attribute* is an array of *RolePermissionType Structures* (see Table 8).

Table 8 – RolePermissionType

Name	Type	Description
RolePermissionType	Structure	Specifies the *Permissions* for a *Role*
roleId	NodeId	The *NodeId* of the *Role Object*.
permissions	PermissionType	A mask specifying which *Permissions* are available to the *Role*.

Servers may allow administrators to write to the *RolePermissions Attribute*.

If not specified, the value of *DefaultRolePermissions Property* from the *NamespaceMetadata Object* associated with the *Node* shall be used instead. If the *NamespaceMetadata Object* does not define the *Property* or does not exist, then the *Server* should not publish any information about how it manages *Permissions*.

If a *Server* supports *Permissions* for a particular *Namespace* it shall add the *DefaultRolePermissions Property* to the *NamespaceMetadata Object* for that *Namespace* (see Figure 8). If a particular *Node* in the *Namespace* needs to override the default values, the *Server* adds the *RolePermissions Attribute* to the *Node.* The *DefaultRolePermissions Property* and *RolePermissions Attribute* shall only be readable by administrators. If a *Server* allows the *Permissions* to be changed these values shall be writeable. If the *Server* allows the *Permissions* to be overridden for a particular *Node* but does not currently have any *Node Permissions* configured, then the value of the *Attribute* shall be an empty array. If the administrator wishes to remove overridden *Permissions*, an empty array shall be written to this *Attribute*. *Servers* shall prevent *Permissions* from being changed in such a way as to render the *Server* inoperable.

If a *Server* publishes information about the *Roles* for a *Namespace* assigned to the current *Session,* it shall add the *DefaultUserRolePermissions Property* to the *NamespaceMetadata Object* for that *Namespace.* The value of this *Property* shall be a readonly list of *Permissions* for each *Role* assigned to the current *Session*. If a particular *Node* in the *Namespace* overrides the default *RolePermissions* the *Server* shall also override the *DefaultUserRolePermissions* by adding the *UserRolePermissions Attribute* to the *Node.* If the *Server* allows the *Permissions* to be overridden for a particular *Node* but does not currently have any *Node Permissions* configured, then the Server shall return the value of the *DefaultUserRolePermissions Property* for the *Node Namespace.*

If a *Server* implements a vendor specific *Role Permission* model for a *Namespace,* it shall not add the *DefaultRolePermissions* or *DefaultUserRolePermissions Properties* to the *Namespace Metadata Object.*

Documents
Document Sharing Hub
Full document details: E DIN EN IEC 62541-3:2019-09