

DIN EN 62541-100

ICS 35.200; 35.240.50

**OPC Unified Architecture –
Teil 100: Geräteschnittstelle
(IEC 62541-100:2015);
Englische Fassung EN 62541-100:2015**

OPC unified architecture –
Part 100: Device Interface
(IEC 62541-100:2015);
English version EN 62541-100:2015

Architecture unifiée OPC –
Partie 100: Interface d'appareils
(IEC 62541-100:2015);
Version anglaise EN 62541-100:2015

Gesamtumfang 60 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE

Anwendungsbeginn

Anwendungsbeginn für die von CENELEC am 2015-04-29 angenommene Europäische Norm als DIN-Norm ist 2015-10-01.

Nationales Vorwort

Vorausgegangener Norm-Entwurf: E DIN EN 62541-100:2014-08.

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

Die enthaltene IEC-Publikation wurde vom SC 65E „Devices and integration in enterprise systems“ erarbeitet.

Das IEC-Komitee hat entschieden, dass der Inhalt dieser Publikation bis zu dem Datum (stability date) unverändert bleiben soll, das auf der IEC-Website unter „<http://webstore.iec.ch>“ zu dieser Publikation angegeben ist. Zu diesem Zeitpunkt wird entsprechend der Entscheidung des Komitees die Publikation

- bestätigt,
- zurückgezogen,
- durch eine Folgeausgabe ersetzt oder
- geändert.

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.

Da sich die Benutzer der vorliegenden Norm der englischen Sprache als Fachsprache bedienen, wird die Englische Fassung der EN 62541-100 veröffentlicht. Zu deren Abschnitt 1, der den Anwendungsbereich festlegt, und Abschnitt 3, der die Begriffe festlegt, wurde eine Übersetzung angefertigt und als Nationaler Anhang NA der vorliegenden Norm hinzugefügt. Für viele der verwendeten Begriffe existieren keine gebräuchlichen deutschen Benennungen, da sich die deutschen Anwender in der Regel ebenfalls der englischen Benennungen bedienen. Diese Norm steht nicht in unmittelbarem Zusammenhang mit Rechtsvorschriften und ist nicht als Sicherheitsnorm anzusehen.

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.

Nationaler Anhang NA

(informativ)

1 Anwendungsbereich

Dieser Teil der Normenreihe IEC 62541 ist eine Erweiterung der gesamten Normenreihe zur OPC Unified Architecture (vereinheitlichte Architektur der plattformunabhängigen Kommunikation) (OPC-UA)) und legt das Informationsmodell fest, welches im Zusammenhang mit *Geräten* (en: *Devices*)) steht. Dieser Teil der Normenreihe IEC 62541 legt drei Modelle fest, welche wie folgt aufeinander aufbauen:

- das (Basis-)Gerätemodell ist vorgesehen für die Bereitstellung einer einheitlichen Sicht auf Geräte, unabhängig von den zugrundeliegenden Geräteprotokollen;
- das Geräte-Kommunikations-Modell ergänzt Informationselemente bezüglich Netzwerk und Verbindung, sodass Kommunikations-Topologien erstellt werden können;
- das Geräte-Integrations-Host-Modell ergänzt schließlich zusätzliche Elemente und Regeln, welche für Host-Systeme erforderlich sind, um die Integration für ein vollständiges System zu handhaben. Es ermöglicht sowohl eine Betrachtung der Topologie des Automatisierungssystems mit den Geräten als auch der verbindenden Kommunikationsnetzwerke.

3 Begriffe, Abkürzungen und verwendete Datentypen

3.1 Begriffe

Für die Anwendung dieses Dokuments gelten die Begriffe nach IEC/TR 62541-1, IEC 62541-3 und IEC 62541-8 und die folgenden Begriffe.

3.1.1

Block

(en: block)

eine funktionale Gruppierungseinheit für *Parameter*

ANMERKUNG 1 zum Begriff: Sie könnte auf einen Funktionsblock (siehe IEC 61131, alle Teile) *Feldgeräteintegration* (en: *Field Device Integration* (FDI)) oder die Ressourcenparameter des Gerätes selbst abgebildet werden.

3.1.2

blockModus

(en: blockMode)

Betriebsmodus für einen *Block* (Zielmodus, zulässige Modi, tatsächlicher Modus und bestimmungsgemäßer Modus).

ANMERKUNG 1 zum Begriff: Weitere Einzelheiten zu *BlockModi* werden von Normungsorganisationen festgelegt.

3.1.3

Kommunikationsprofil

(en: Communication Profile)

festgelegter Satz von Abbildungsregeln zur Ermöglichung einer eindeutigen Interoperabilität zwischen *Geräten* bzw. Anwendungen

ANMERKUNG 1 zum Begriff: Beispiele für derartige Profile sind die „kabellosen Kommunikationsnetze und Kommunikationsprofile für WirelessHART“ in IEC 62591 und die Protokollabbildungen für OPC-UA in IEC 62541-6.

3.1.4

Verbindungspunkt

(en: Connection Point)

logische Darstellung der Schnittstelle zwischen einem *Gerät* und einem *Netz*

3.1.5

Gerät

(en: Device)

unabhängige physikalische Einheit, die fähig ist, eine oder mehrere spezifische Funktionen in einem speziellen Kontext auszuführen und die durch ihre Schnittstellen abgegrenzt ist

ANMERKUNG 1 zum Begriff: Siehe IEC 61499-1.

ANMERKUNG 2 zum Begriff: *Geräte* stellen Funktionen zur Abtastung, Schaltung, Kommunikation und/oder Steuerung bereit. Beispiele sind Messumformer, Ventilsteuergeräte, Getriebe, Motorregler, SPS und Kommunikations-Gateways.

3.1.6

Geräteintegrationshost

(en: Device Integration Host)

Server, der für die Integration mehrerer *Geräte* in ein Automatisierungssystem sorgt

3.1.7

Gerätetopologie

(en: Device Topology)

Anordnung von *Netzen* und *Geräten*, die eine Kommunikationstopologie bildet

3.1.8

Feldbus

(en: fieldbus)

Kommunikationssystem basierend auf serieller Datenübertragung und eingesetzt in der industriellen Automatisierung oder in Prozessleitsystemen

ANMERKUNG 1 zum Begriff: Siehe IEC 61784.

ANMERKUNG 2 zum Begriff: Bezeichnet den Kommunikationsbus, der von einem Gerät verwendet wird.

3.1.9

Parameter

(en: Parameter)

Variable des *Gerätes*, die für Konfigurations-, Überwachungs- und Steuerungszwecke verwendet werden kann

ANMERKUNG 1 zum Begriff: Im Informationsmodell ist er gleichbedeutend mit einer OPC-UA-*DatenVariable*

3.1.10

Netz

(en: Network)

stellt ein Mittel zur Kommunikation mit einem bestimmten Protokoll dar

English Version

OPC unified architecture - Part 100: Device Interface
(IEC 62541-100:2015)

Architecture unifiée OPC - Partie 100: Interface d'appareils
(IEC 62541-100:2015)

OPC Unified Architecture - Teil 100: Geräteschnittstelle
(IEC 62541-100:2015)

This European Standard was approved by CENELEC on 2015-04-29. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

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

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

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



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

CEN-CENELEC Management Centre: Avenue Marnix 17, B-1000 Brussels

Foreword

The text of document 65E/372/CDV, future edition 1 of IEC 62541-100, prepared by SC 65E "Devices and integration in enterprise systems", of IEC/TC 65 "Industrial-process measurement, control and automation" was submitted to the IEC-CENELEC parallel vote and approved by CENELEC as EN 62541-100:2015.

The following dates are fixed:

- latest date by which the document has to be implemented at national level by publication of an identical national standard or by endorsement (dop) 2016-01-29
- latest date by which the national standards conflicting with the document have to be withdrawn (dow) 2018-04-29

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

This document has been prepared under a mandate given to CENELEC by the European Commission and the European Free Trade Association, and supports essential requirements of EU Directive(s).

Endorsement notice

The text of the International Standard IEC 62541-100:2015 was approved by CENELEC as a European Standard without any modification.

In the official version, for Bibliography, the following notes have to be added for the standards indicated:

IEC 61131	NOTE	Harmonized in EN 61131 series (not modified).
IEC 61499-1:2012	NOTE	Harmonized as EN 61499-1:2013 (not modified).
IEC 61784	NOTE	Harmonized in EN 61784 series (not modified).
IEC 62591	NOTE	Harmonized as EN 62591.
IEC 62769	NOTE	Harmonized in EN 62769 series ¹⁾ (not modified).

1) At draft stage.

CONTENTS

1	Scope.....	7
2	Reference documents	7
3	Terms, definitions, abbreviations and used data types	7
3.1	Terms and definitions	7
3.2	Abbreviations	9
3.3	Used data types	9
4	Fundamentals.....	9
4.1	OPC UA.....	9
4.2	Conventions used in this document.....	10
4.2.1	Conventions for Node descriptions	10
4.2.2	NodeIds and BrowseNames	11
5	Device model.....	12
5.1	General.....	12
5.2	TopologyElementType.....	13
5.3	FunctionalGroupType	15
5.4	Identification FunctionalGroup	17
5.5	UIElement Type	18
5.6	DeviceType.....	19
5.7	Device support information	21
5.7.1	General	21
5.7.2	Device Type Image	22
5.7.3	Documentation.....	22
5.7.4	Protocol support files	22
5.7.5	Images	23
5.8	DeviceSet entry point	23
5.9	ProtocolType.....	24
5.10	BlockType.....	25
5.11	Configurable components.....	27
5.11.1	General pattern.....	27
5.11.2	ConfigurableObjectType.....	27
6	Device vomunication model.....	28
6.1	General.....	28
6.2	Network	29
6.3	ConnectionPoint.....	30
6.4	ConnectsTo and ConnectsToParent ReferenceTypes.....	32
6.5	NetworkSet Object (mandatory)	33
7	Device integration host model	34
7.1	General.....	34
7.2	DeviceTopology Object	35
7.3	Online/Offline.....	36
7.3.1	General	36
7.3.2	IsOnline ReferenceType	37
8	AddIn Capabilities	38
8.1	Overview	38

8.2	Offline-Online data transfer	39
8.2.1	Definition	39
8.2.2	TransferServices Type	39
8.2.3	TransferServices Object.....	40
8.2.4	TransferToDevice Method	40
8.2.5	TransferFromDevice Method	41
8.2.6	FetchTransferResultData Method	42
8.3	Locking.....	44
8.3.1	Overview	44
8.3.2	LockingServices Type	44
8.3.3	LockingServices Object.....	46
8.3.4	MaxInactiveLockTime Property.....	46
8.3.5	InitLock Method	46
8.3.6	ExitLock Method	47
8.3.7	RenewLock Method	48
8.3.8	BreakLock Method	48
9	Specialized topology elements.....	49
9.1	General	49
9.2	Block Devices (BlockOriented DeviceType)	49
9.3	Modular Devices	50
10	Profiles	51
10.1	General	51
10.2	Device Server Facets.....	51
10.3	Device Client Facets	52
Annex A (normative) Namespace and mappings.....		54
Bibliography		55
Annex ZA (normative) Normative references to international publications with their corresponding European publications.....		56
Figure 1 – Device model overview.....		12
Figure 2 – Components of the TopologyElementType		13
Figure 3 – FunctionalGroupType		15
Figure 4 – Analyser Device use for FunctionalGroups (UA Companion ADI)		16
Figure 5 – PLCopen use for FunctionalGroups (UA Companion PLCopen)		17
Figure 6 – Example of an Identification FunctionalGroup		18
Figure 7 – DeviceType.....		19
Figure 8 – Integration of support information within a DeviceType.....		21
Figure 9 – Standard entry point for Devices.....		24
Figure 10 – Example of a ProtocolType hierarchy with instances that represent specific communication profiles		25
Figure 11 – BlockType hierarchy		26
Figure 12 – Configurable component pattern		27
Figure 13 – ConfigurableObjectType		28
Figure 14 – Initial example of a communication topology		29
Figure 15 – NetworkType		29

Figure 16 – Example of ConnectionPointType hierarchy	30
Figure 17 – ConnectionPointType	31
Figure 18 – ConnectionPoint usage	32
Figure 19 – Type hierarchy for ConnectsTo and ConnectsToParent References	32
Figure 20 – Example with ConnectsTo and ConnectsToParent References	33
Figure 21 – Example of an automation system	34
Figure 22 – Example of a Device topology	35
Figure 23 – Online component for access to device data	36
Figure 24 – Type hierarchy for <i>IsOnline Reference</i>	38
Figure 25 – TransferServicesType	39
Figure 26 – TransferServices	40
Figure 27 – LockingServicesType	45
Figure 28 – LockingServices	46
Figure 29 – Block-oriented Device structure example	49
Figure 30 – Modular Device structure example	50
Table 1 – DataTypes defined in IEC 62541-3	9
Table 2 – Type definition table	10
Table 3 – Examples of DataTypes	11
Table 4 – TopologyElementType definition	14
Table 5 – ParameterSet definition	14
Table 6 – MethodSet definition	14
Table 7 – FunctionalGroupType definition	15
Table 8 – UIElementType definition	18
Table 9 – DeviceType definition	19
Table 10 – DeviceHealth values	21
Table 11 – DeviceTypeImage definition	22
Table 12 – Documentation definition	22
Table 13 – ProtocolSupport definition	22
Table 14 – ImageSet definition	23
Table 15 – DeviceSet definition	24
Table 16 – ProtocolType definition	25
Table 17 – BlockType definition	26
Table 18 – ConfigurableObjectType definition	28
Table 19 – NetworkType definition	30
Table 20 – ConnectionPointType definition	31
Table 21 – ConnectsTo ReferenceType	33
Table 22 – ConnectsToParent ReferenceType	33
Table 23 – NetworkSet definition	33
Table 24 – DeviceTopology definition	36
Table 25 – <i>IsOnline ReferenceType</i>	38
Table 26 – TransferServicesType definition	39

Table 27 – TransferToDevice Method arguments	41
Table 28 – TransferToDevice Method AddressSpace definition	41
Table 29 – TransferFromDevice Method arguments	41
Table 30 – TransferFromDevice Method AddressSpace definition	41
Table 31 – FetchTransferResultData Method Arguments	43
Table 32 – FetchTransferResultData Method AddressSpace definition	43
Table 33 – FetchResultDataType structure	43
Table 34 – TransferResultError DataType structure	43
Table 35 – TransferResultData DataType structure	44
Table 36 – LockingServicesType definition	45
Table 37 – MaxInactiveLockTime Property definition	46
Table 38 – InitLock Method Arguments	47
Table 39 – InitLock Method AddressSpace definition	47
Table 40 – ExitLock Method Arguments	48
Table 41 – ExitLock Method AddressSpace definition	48
Table 42 – RenewLock Method Arguments	48
Table 43 – RenewLock Method AddressSpace definition	48
Table 44 – BreakLock Method Arguments	49
Table 45 – BreakLock Method AddressSpace definition	49
Table 46 – BaseDevice_Server_Facet definition	51
Table 47 – DeviceIdentification_Server_Facet definition	51
Table 48 – BlockDevice_Server_Facet definition	51
Table 49 – Locking_Server_Facet definition	51
Table 50 – DeviceCommunication_Server_Facet definition	52
Table 51 – DeviceIntegrationHost_Server_Facet definition	52
Table 52 – BaseDevice_Client_Facet definition	52
Table 53 – DeviceIdentification_Client_Facet definition	52
Table 54 – BlockDevice_Client_Facet definition	52
Table 55 – Locking_Client_Facet definition	53
Table 56 – DeviceCommunication_Client_Facet definition	53
Table 57 – DeviceIntegrationHost_Client_Facet definition	53

OPC UNIFIED ARCHITECTURE –

Part 100: Device Interface

1 Scope

This part of IEC 62541 is an extension of the overall OPC Unified Architecture standard series and defines the information model associated with *Devices*. This part of IEC 62541 describes three models which build upon each other as follows:

- the (base) Device Model is intended to provide a unified view of devices irrespective of the underlying device protocols;
- the Device Communication Model adds Network and Connection information elements so that communication topologies can be created;
- the Device Integration Host Model finally adds additional elements and rules required for host systems to manage integration for a complete system. It allows reflecting the topology of the automation system with the devices as well as the connecting communication networks.

2 Reference documents

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

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

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

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

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

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

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

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

NAMUR Recommendation NE107: *Self-monitoring and diagnosis of field devices*

3 Terms, definitions, abbreviations and used data types

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC TR 62541-1, IEC 62541-3, and IEC 62541-8 as well as the following apply.

3.1.1

block

functional *Parameter* grouping entity

Note 1 to entry: It could map to a function block (see IEC 62769 (all parts), *Field Device Integration (FDI)*)

NOTE Harmonized in EN 62769 series) (not modified).

) or to the resource parameters of the device itself.

3.1.2

blockMode

mode of operation (target mode, permitted modes, actual mode, and normal mode) for a *Block*

Note 1 to entry: Further details about *Block* modes are defined by standard organisations.

3.1.3

Communication Profile

fixed set of mapping rules to allow unambiguous interoperability between *Devices* or *Applications*, respectively

Note 1 to entry: Examples of such profiles are the “Wireless communication network and communication profiles for WirelessHART” in IEC 62591 and the Protocol Mappings for OPC UA in IEC 62541-6.

3.1.4

Connection Point

logical representation of the interface between a *Device* and a *Network*

3.1.5

device

independent physical entity capable of performing one or more specified functions in a particular context and delimited by its interfaces

Note 1 to entry: See IEC 61499-1.

Note 2 to entry: *Devices* provide sensing, actuating, communication, and/or control functionality. Examples include transmitters, valve controllers, drives, motor controllers, PLCs, and communication gateways.

3.1.6

Device Integration Host

Server that manages integration of multiple *Devices* in an automation system

3.1.7

Device Topology

arrangement of *Networks* and *Devices* that constitute a communication topology

3.1.8

fieldbus

communication system based on serial data transfer and used in industrial automation or process control applications

Note 1 to entry: See IEC 61784.

Note 2 to entry: Designates the communication bus used by a *Device*.

3.1.9

parameter

variable of the *Device* that can be used for configuration, monitoring or control purposes

Note 1 to entry: In the information model it is synonymous to an OPC UA *DataVariable*.

3.1.10

network

means used to communicate with one specific protocol

3.2 Abbreviations

ADI	Analyser Device Integration
CP	Communication Processor (hardware module)
CPU	Central Processing Unit (of a <i>Device</i>)
DA	Data Access
DI	Device Integration (the short name for this standard)
UA	Unified Architecture
UML	Unified Modelling Language
XML	Extensible Mark-up Language

3.3 Used data types

Table 1 describes the *DataTypes* that are used throughout this document.

Table 1 – DataTypes defined in IEC 62541-3

Parameter Type
Argument
Boolean
Duration
LocalizedText
String
Int32

4 Fundamentals

4.1 OPC UA

The main use case for OPC standards is the online data exchange between devices and HMI or SCADA systems using Data Access functionality. In this use case the device data is provided by an OPC *Server* and is consumed by an OPC *Client* integrated into the HMI or SCADA system. OPC DA provides functionality to browse through hierarchical namespaces containing data items and to read, write and to monitor these items for data changes. The classic OPC standards are based on Microsoft COM/DCOM technology for the communication between software components from different vendors. Therefore classic OPC *Server* and *Clients* are restricted to Windows PC based automation systems.

OPC UA incorporates all features of classic OPC standards like OPC DA, A&E and HDA but defines platform independent communication mechanisms and generic, extensible and object-oriented modelling capabilities for the information a system wants to expose.

The OPC UA network communication part defines different mechanisms optimized for different use cases. The current version of OPC UA is defining an optimized binary protocol for high performance intranet communication as well as Web Services. It allows adding new protocols in the future. Features like security, access control and reliability are directly built into the transport mechanisms. Based on the platform independence of the protocols, OPC UA *Servers* and *Clients* can be directly integrated into devices and controllers.

The OPC UA *Information Model* provides a standard way for *Servers* to expose *Objects* to *Clients*. *Objects* in OPC UA terms are composed of other *Objects*, *Variables* and *Methods*. OPC UA also allows relationships to other *Objects* to be expressed.

The set of *Objects* and related information that an OPC UA *Server* makes available to *Clients* is referred to as its *AddressSpace*. The elements of the OPC UA *Object Model* are represented in the *AddressSpace* as a set of *Nodes* described by *Attributes* and interconnected by *References*. OPC UA defines eight classes of *Nodes* to represent *AddressSpace* components. The classes are *Object*, *Variable*, *Method*, *ObjectType*, *VariableType*, *DataTypes*, *ReferenceType* and *View*. Each *NodeClass* has a defined set of *Attributes*.

This standard makes use of almost all OPC UA *NodeClasses*.

Objects are used to represent real-world entities such as *Devices* and (communication) *Networks* as well as software entities such as *Blocks*. An *Object* is associated to a corresponding *ObjectType* that provides definitions for that *Object*.

Variables are used to represent values. Two categories of *Variables* are defined, *Properties* and *DataVariables*.

Properties are *Server*-defined characteristics of *Objects*, *DataVariables* and other *Nodes*. *Properties* are not allowed to have *Properties* defined for them. Examples for *Properties* of *Objects* are the device serial number and the block tag.

DataVariables represent the contents of an *Object*. *DataVariables* may have component *DataVariables*. This is typically used by *Servers* to expose individual elements of arrays and structures. This standard uses *DataVariables* to represent the *Parameters* of both *Blocks* and *Devices*.

4.2 Conventions used in this document

4.2.1 Conventions for Node descriptions

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

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	Data Type	Type Definition	Modelling Rule
Reference Type name	NodeClass of the TargetNode.	BrowseName of the target Node. If the Reference is to be instantiated by the server, then the value of the target Node's BrowseName is "--".	Attributes of the referenced Node, only applicable for Variables and Objects.		Referenced ModellingRule of the referenced Object.
NOTE Notes referencing footnotes of the table content.					

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 *DataTypes* 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}”. The *ValueRank* is set to the corresponding value (see IEC 62541-3) and the *ArrayDimensions* is set to null or is omitted. In Table 3 examples are given.

Table 3 – Examples of DataTypes

Notation	Data-Type	Value-Rank	ArrayDimensions	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 *NodeId* of a *TypeDefinitionNode*, i.e. the specified *Node* points with a *HasTypeDefinition Reference* to the corresponding *TypeDefinitionNode*. The symbolic name of the *NodeId* is used in the table.
- 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.

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.

Components of *Nodes* can be complex, i.e. contain 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 4.2.2.1. Therefore those containing components are not explicitly specified; they are implicitly specified by the type definitions.

4.2.2 NodeIds and BrowseNames

4.2.2.1 NodeIds

The *NodeIds* of all *Nodes* described in this document are only symbolic names. Annex A defines the actual *NodeIds*.

The symbolic name of each *Node* defined in this document is its *BrowseName*, or, when it is part of another *Node*, the *BrowseName* of the other *Node*, a “.”, and the *BrowseName* of itself. In this case “part of” means that the whole has a *HasProperty* or *HasComponent Reference* to its part. Since all *Nodes* not being part of another *Node* have a unique name in this document, the symbolic name is unique.

The namespace for this standard is defined in Annex A. The *NamespaceIndex* for all *NodeIds* and *BrowseNames* defined in this standard is server specific and depends on the position of the namespace URI in the server namespace table.

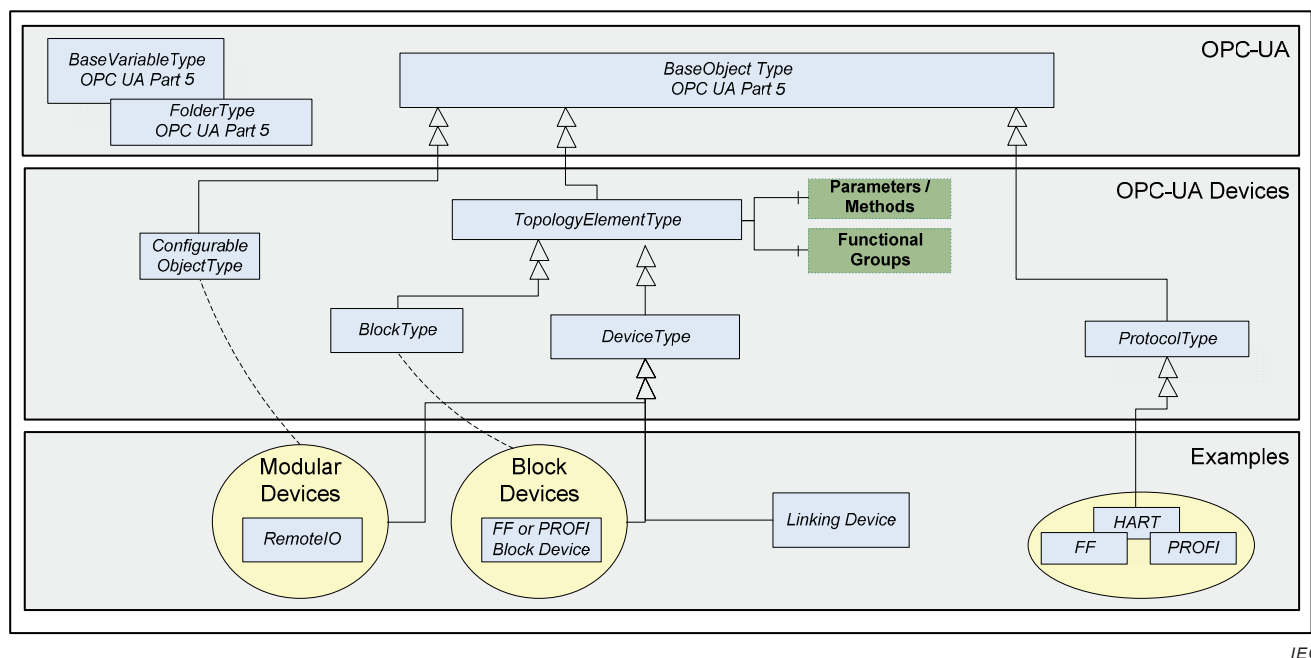
4.2.2.2 BrowseNames

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 server specific and depends on the position of the namespace URI in the server namespace table.

5 Device model

5.1 General

Figure 1 depicts the main *ObjectTypes* of the base *Device* model and their relationship. The drawing is not intended to be complete. For the sake of simplicity only a few components and relations were captured so as to give a rough idea of the overall structure.



IEC

Figure 1 – Device model overview

The boxes in this drawing show the *ObjectTypes* used in this standard as well as some elements from other standards that help understand some modelling decisions. The upper grey box shows the OPC UA core *ObjectTypes* from which the *TopologyElement Type* and *ProtocolType* are derived. The grey box in the second level shows the main *ObjectTypes* that this standard introduces. The components of those *ObjectTypes* are illustrated only in an abstract way in this overall picture.

The grey box in the third level shows real-world examples as they will be used in products and plants. In general, such subtypes are defined by other organizations.

The *TopologyElement Type* is the base *ObjectType* for elements in a device topology. It introduces *Parameters* and *Methods*. This standard also defines a functional grouping concept to provide alternative viewpoints.

The *DeviceType ObjectType* provides a general type definition for any *Device*. *Devices* – in addition to *Parameters* and *Methods* – may support subdevices and may support *Blocks*. *Blocks* are typically used as means to organise the functionality within a *Device*. Specific types of *Blocks* will be specified by the various field communication foundations.

A *ProtocolType ObjectType* represents a specific communication/*FieldBus* protocol implemented by a certain *TopologyElement*. Examples are *FFBusType*, *HARTBusType*, or *PROFIBUS/PROFINETType*.

The *ConfigurableObjectType* is used as a general means to create modular topology units. If needed an instance of this type will be added to the head object of the modular unit. Modular *Devices*, for example, will use this *ObjectType* to organise their *Modules*. Block-oriented *Devices* use it to expose and organise their *Blocks*.

5.2 TopologyElementType

This *ObjectType* defines the basic information components for all configurable elements in a device topology. It introduces *ParameterSet*, *MethodSet*, and *FunctionalGroups*. Figure 2 shows the *TopologyElementType*. It is formally defined in Table 4.

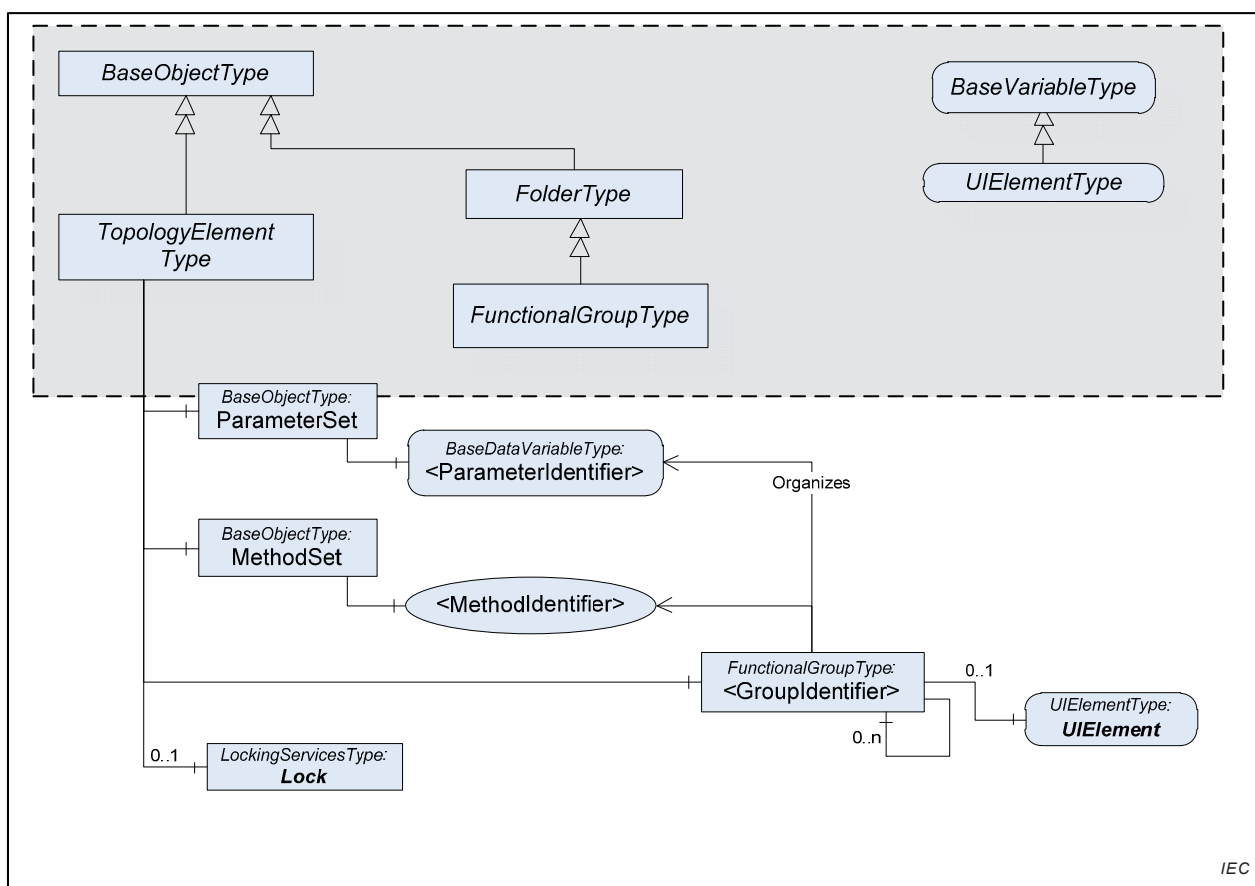


Figure 2 – Components of the TopologyElementType

All elements in a topology may have *Parameters* and *Methods*. If such an element has *Parameters* they are kept in an *Object* called “*ParameterSet*” as a flat list of *Parameters*. If it has *Methods* they are kept the same way in an *Object* called “*MethodSet*”. *FunctionalGroups* can be used to organise the *Parameters* and *Methods* to reflect the structure of the *TopologyElement*. The same *Parameter* or *Method* might be referenced from more than one *FunctionalGroup*. Each group may reference an optional user interface for the represented functionality. *FunctionalGroups* are specified in 5.3.

- Rule for all *Methods* organised in *FunctionalGroups*:

Since *Methods* are components of the *MethodSet Object*, *Clients* have to specify the *NodeId* of the *MethodSet* instance in the *ObjectId* argument of the *Call Service*.

Parameters are modelled with OPC UA *DataVariable* nodes. A *Parameter* can be an instance of *BaseDataVariableType* (defined in IEC 62541-5) or any sub-type defined by OPC UA or other standard organisations. Some of the most common *VariableTypes* (*DataItemTypes*, *AnalogItemTypes*, and *DiscreteItemTypes*) are defined in IEC 62541-8. In this standard, the term *Parameter* is used generically – regardless of *VariableType*.

The *TopologyElement ObjectType* is formally defined in Table 4

Table 4 – TopologyElementType definition

Attribute	Value				
BrowseName	TopologyElementType				
IsAbstract	True				
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
Subtype of the <i>BaseObjectType</i> defined in IEC 62541-5					
HasSubtype	ObjectType	DeviceType	Defined in 5.3		
HasSubtype	ObjectType	BlockType	Defined in 5.10		
HasComponent	Object	ParameterSet		BaseObjectType	Optional
HasComponent	Object	MethodSet		BaseObjectType	Optional
HasComponent	Object	<GroupIdentifier>		FunctionalGroupType	OptionalPlaceholder
HasComponent	Object	Identification		FunctionalGroupType	Optional
HasComponent	Object	Lock		LockingServicesType	Optional

The *TopologyElementType* is abstract. There will be no instances of a *TopologyElementType* itself, but there will be instances of sub-types of this type. In this standard, the term *TopologyElement* generically refers to an instance of any *ObjectType* derived from the *TopologyElementType*.

ParameterSet gathers the references to all *Parameters* that are exposed to the *Client*. The *ParameterSet* is mandatory if *Parameters* exist for a *TopologyElement*. The “*ParameterSet*” *Object* is formally defined in Table 5.

Table 5 – ParameterSet definition

Attribute	Value			
BrowseName	ParameterSet			
References	NodeClass	BrowseName	TypeDefinition	ModellingRule
HasTypeDefinition	ObjectType	BaseObjectType		
HasComponent	Variable	<ParameterIdentifier>	BaseDataVariableType	MandatoryPlaceholder

MethodSet gathers the references to all *Methods* that are exposed to the *Client*. The “*MethodSet*” *Object* is formally defined in Table 6.

Table 6 – MethodSet definition

Attribute	Value			
BrowseName	MethodSet			
References	NodeClass	BrowseName	TypeDefinition	ModellingRule
HasTypeDefinition	ObjectType	BaseObjectType		
HasComponent	Method	<MethodIdentifier>		MandatoryPlaceholder