IEC 62453-71

Edition 1.0 2023-11

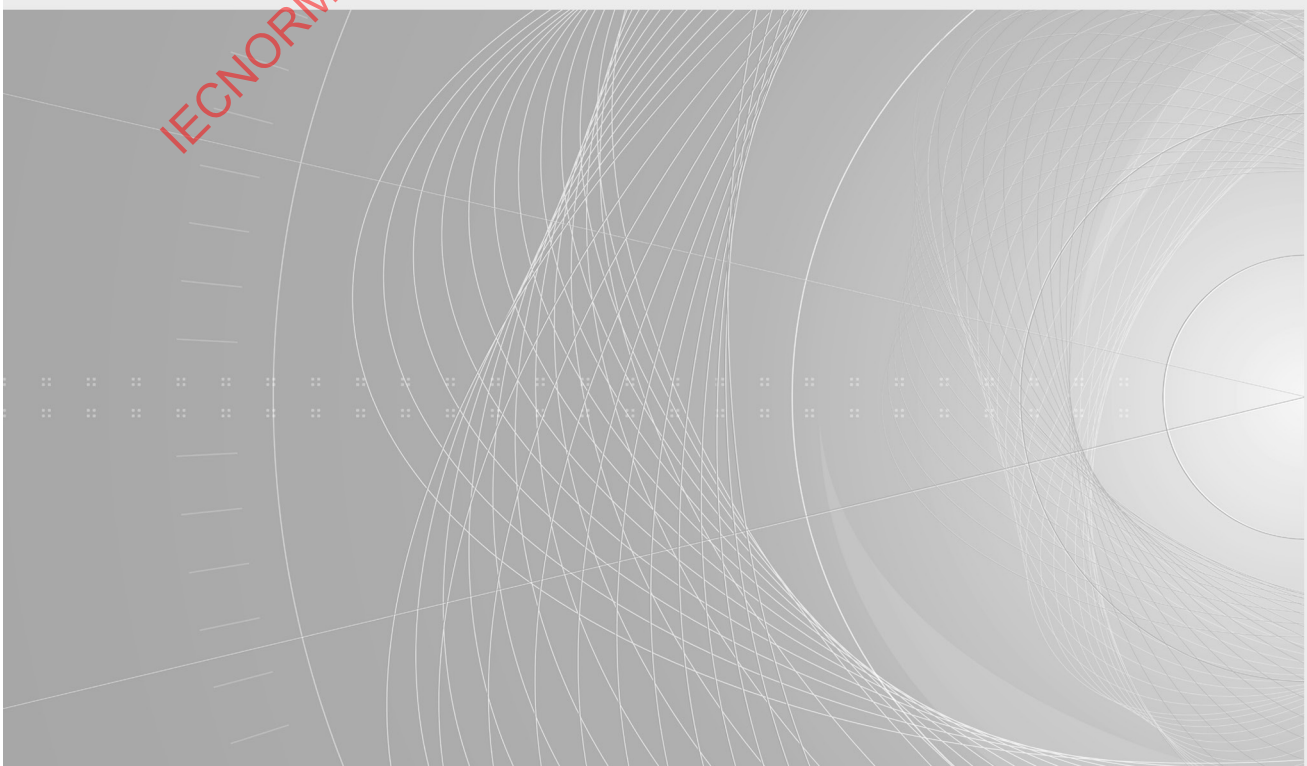
INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Field device tool (FDT) interface specification – Part 71: OPC UA Information Model for FDT

Spécification des interfaces des outils des dispositifs de terrain (FDT) – Partie 71: Modèle d'information de l'OPC UA pour outils FDT



IEC 62453-71:2023-11(en-fr)

IEC Secretariat
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

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

IEC 62453-71

Edition 1.0 2023-11

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Field device tool (FDT) interface specification – Part 71: OPC UA Information Model for FDT

Spécification des interfaces des outils des dispositifs de terrain (FDT) – Partie 71: Modèle d'information de l'OPC UA pour outils FDT

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

ICS 25.040

ISBN 978-2-8322-7619-8

CONTENTS

FOREWORD 7
INTRODUCTION 9
0.1 General 9
0.2 Presentation of FDT 9
0.3 Presentation of OPC Unified Architecture 9
0.4 Presentation of OPC UA Device Integration 10
1 Scope 12
2 Normative references 12
3 Terms, definitions and abbreviated terms 12
3.1 Terms and definitions 12
3.2 Abbreviated terms 13
4 Conventions used in this document 13
4.1.1 Document conventions 13
4.1.2 Conventions for FDT methods 13
4.1.3 Conventions for Node descriptions 13
4.1.4 NodeIds and BrowseNames 16
4.1.5 Common Attributes 17
4.1.6 Graphical notation 19
5 Concept 20
5.1 System architecture 20
6 FDT specific OPC UA ObjectTypes 21
6.1 General 21
6.2 FdtDeviceType 21
6.3 FdtFunctionalGroupType 23
6.4 IFdtDeviceHealthType interface 23
6.5 IFdtSupportInfoType interface 23
6.5.1 Overview 23
6.6 Document types 24
6.6.1 FdtDocumentType 24
6.6.2 FdtDocumentFile 24
6.6.3 FdtDocumentUrl 25
6.7 FdtProtocolType 25
6.8 FdtTransferServiceType 26
6.9 FdtIoSignalInfoType 26
7 OPC UA EventTypes 27
7.1 Overview 27
7.2 FdtAuditEventType 28
7.3 FdtStartMethodEventType 28
7.4 FdtEndMethodEventType 28
7.5 FdtAuditWriteUpdateEventType 29
8 OPC UA VariableTypes 29
8.1 FdtParameter 29
9 OPC UA DataTypes 31
9.1 DataRefType 31
9.2 FdtDeviceClassificationType 31
9.3 SemanticInfoType 32

9.4 Enumeration datatypes 32
9.4.1 AlarmType 32
9.4.2 ApplicationIdEnumeration 33
9.4.3 ClassificationDomainId 33
9.4.4 ClassificationId 34
9.4.5 DocumentClassification 36
9.4.6 FunctionExecutionResultState 36
9.4.7 IECDatatype 37
9.4.8 RangeType 38
9.4.9 SignalTypeEnumeration 38
9.4.10 SubstitutionType 38
9.4.11 SupportedTransfer 39
10 OPC UA ReferenceTypes – HasIOSignalRef 39
11 Mapping of DataTypes 40
11.1 Primitive data types – DeviceHealthEnumeration 40
11.2 Mapping to OPC DI types 40
11.2.1 Device type 40
11.2.2 TopologyElementType 45
11.2.3 FunctionalGroupType 46
11.2.4 Identification FunctionalGroup 47
11.2.5 Device data and device methods 48
11.2.6 Methods 49
11.2.7 Variable 52
11.2.8 Device support information 54
11.2.9 FdtProtocolType 56
12 Profiles and Conformance Units 56
12.1 Conformance Units 56
12.2 Profiles 57
12.2.1 Profile list 57
12.2.2 Server Facets 57
12.2.3 Client Facets 58
13 Namespaces 59
13.1 Namespace metadata 59
13.2 Handling of OPC UA namespaces 60
Annex A (normative) FDT namespace and identifiers 61
Annex B (informative) Use cases 62
B.1 General 62
B.2 Use case: List topology 62
B.3 Use case: Identify device 63
B.4 Get list of available device parameters 64
B.4.1 Use case: Browse device parameters 64
B.4.2 Use case: Get attributes of a device parameter 65
B.5 Use case: Get Device Status 66
B.6 Use case: Get Device Diagnostics 67
B.7 Read parameters 68
B.7.1 Use case: Read offline data 68
B.7.2 Use case: Read online data 69
B.8 Use case: Write device parameters 70

B.9 Use case: Audit trail 71
Bibliography 72

Figure 1 – OPC UA Devices Example 11
Figure 2 – The OPC UA Information Model Notation 19
Figure 3 – System architecture according to IEC 62453-42 21
Figure 4 – FdtDeviceType overview 22
Figure 5 – FdtProtocolType overview 25
Figure 6 – FdtTransferServiceType overview 26
Figure 7 – FdtIoSignalInfoType overview 27
Figure 8 – Audit event type overview 28
Figure 9 – Example for sources of DeviceType information 41
Figure 10 – Example for sources of TopologyType information 45
Figure 11 – Example for mapping of data and function information 49
Figure 12 – Example for source of function information 50
Figure 13 – Example for source of static function information 51

Table 1 – Examples of DataTypes 14
Table 2 – Example for type definition 15
Table 3 – Examples of other characteristics 15
Table 4 – <some>Type Additional References 15
Table 5 – <some>Type Additional sub-components 16
Table 6 – <some>Type Attribute values for child Nodes 16
Table 7 – Common Node Attributes 17
Table 8 – Common Object Attributes 18
Table 9 – Common Variable Attributes 18
Table 10 – Common VariableType Attributes 18
Table 11 – Common Method Attributes 19
Table 12 – FdtDeviceType definition 22
Table 13 – FdtFunctionalGroupType definition 23
Table 14 – IFdtDeviceHealthType definition 23
Table 15 – IFdtSupportInfoType definition 24
Table 16 – IFdtSupportInfoType additional subcomponents 24
Table 17 – FdtDocumentType definition 24
Table 18 – FdtDocumentFile definition 25
Table 19 – FdtDocumentUrl definition 25
Table 20 – FdtProtocolType definition 26
Table 21 – FdtTransferServiceType definition 26
Table 22 – FdtIoSignalInfoType definition 27
Table 23 – FdtAuditEventType definition 28
Table 24 – FdtStartMethodEventType definition 28
Table 25 – FdtEndMethodEventType definition 29
Table 26 – FdtAuditWriteUpdateEventType definition 29

Table 27 – FdtParameter definition 30
Table 28 – DataRefType structure 31
Table 29 – DataRefType definition 31
Table 30 – FdtDeviceClassificationType structure 31
Table 31 – FdtDeviceClassificationType definition 32
Table 32 – SemanticInfoType structure 32
Table 33 – SemanticInfoType definition 32
Table 34 – AlarmType items 32
Table 35 – AlarmType definition 33
Table 36 – ApplicationIdEnumeration items 33
Table 37 – ApplicationIdEnumeration definition 33
Table 38 – ClassificationDomainId items 34
Table 39 – ClassificationDomainId definition 34
Table 40 – ClassificationId items 34
Table 41 – ClassificationId definition 36
Table 42 – DocumentClassification items 36
Table 43 – DocumentClassification definition 36
Table 44 – FunctionExecutionResultState items 36
Table 45 – FunctionExecutionResultState definition 37
Table 46 – IECDatatype items 37
Table 47 – IECDatatype definition 37
Table 48 – RangeType items 38
Table 49 – RangeType definition 38
Table 50 – SignalTypeEnumeration items 38
Table 51 – SignalTypeEnumeration definition 38
Table 52 – SubstitutionType items 39
Table 53 – SubstitutionType definition 39
Table 54 – SupportedTransfer items 39
Table 55 – SupportedTransfer definition 39
Table 56 – HasIOSignalRef definition 40
Table 57 – Mapping for DeviceHealthEnumeration 40
Table 58 – DeviceType mapping 42
Table 59 – Device information mapping 43
Table 60 – Offline device parameter mapping 44
Table 61 – Online device parameter mapping 44
Table 62 – TopologyElementType mapping 46
Table 63 – FunctionalGroupType mapping 47
Table 64 – Mapping for FunctionalGroup Identification 47
Table 65 – Method node information mapping 50
Table 66 – Method node information mapping for static function 51
Table 67 – TransferService mapping 52
Table 68 – Mapping of FDT data items 52
Table 69 – FdtParameter mapping 53

Table 70 – Mapping of simple data types 54
Table 71 – Device Type Image mapping 55
Table 72 – ProtocolSupport mapping 55
Table 73 – FdtIoSignalInfoType node information mapping 56
Table 74 – FdtProtocolType node information mapping 56
Table 75 – Conformance Units for FDT 57
Table 76 – Profile URIs for FDT 57
Table 77 – FDT Base Server Profile 58
Table 78 – FDT General Server Facet 58
Table 79 – FDT General Client Facet 59
Table 80 – NamespaceMetadata Object for this document 59
Table 81 – Namespaces used in a FDT Server 60
Table 82 – Namespaces used in this document 60

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

FIELD DEVICE TOOL (FDT) INTERFACE SPECIFICATION –

Part 71: OPC UA Information Model for FDT

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) IEC draws attention to the possibility that the implementation of this document may involve the use of (a) patent(s). IEC takes no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, IEC had not received notice of (a) patent(s), which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at https://patents.iec.ch. IEC shall not be held responsible for identifying any or all such patent rights.

IEC 62453-71 has been prepared by subcommittee 65E: Devices and integration in enterprise systems, of IEC technical committee 65: Industrial-process measurement, control and automation. It is an International Standard.

The text of this International Standard is based on the following documents:

Draft	Report on voting
65E/806/CDV	65E/897A/RVC

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

The language used for the development of this International Standard is English.

This document was drafted in accordance with ISO/IEC Directives, Part 2, and developed in accordance with ISO/IEC Directives, Part 1 and ISO/IEC Directives, IEC Supplement, available at www.iec.ch/members_experts/refdocs. The main document types developed by IEC are described in greater detail at www.iec.ch/standardsdev/publications.

A list of all parts in the IEC 62453 series, published under the general title *Field device tool (FDT) interface specification*, can be found on the IEC website.

The committee has decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under webstore.iec.ch in the data related to the specific document. At this date, the document will be

- reconfirmed,
- withdrawn, or
- revised.

IMPORTANT – The "colour inside" logo on the cover page of this document 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.

INTRODUCTION

0.1 General

The new OPC Unified Architecture (OPC UA) unifies the existing standards and brings them to state-of-the-art technology using service-oriented architecture (SOA). Platform-independent technology allows the deployment of OPC UA beyond current OPC applications only running on Windows-based PC systems. OPC UA can also run on embedded systems as well as Linux / UNIX based enterprise systems. The provided information can be generically modelled and therefore arbitrary information models can be provided using OPC UA.

FDT standardizes the communication and configuration interface between all field devices and host systems. FDT provides a common environment for accessing the devices' most sophisticated features. Any device can be configured, operated, and maintained through the standardized user interface – regardless of supplier, type or communication protocol.

This document specifies a synergy of both approaches, thus allowing easy, standardized access via OPC UA interfaces to device know-how provided on base of FDT.

0.2 Presentation of FDT

FDT is a technology supporting the data exchange between field devices and automation systems. The technology is based on an interface specification standardized as IEC 62453. The specification defines two main concepts: Device Type Manager (DTM) and Frame Application. A DTM is a software component specific to a field device type. A Frame Application is a software environment (part of the automation system) for integration of DTMs. Within a Frame Application every DTM provides data and services specific to the respective field device. Since the technology is based on a standardized set of interfaces, every DTM may be integrated in every Frame Application. Based on FDT it is possible to integrate communication devices, communication infrastructure devices (e.g. gateways) and field devices, depending on their communication protocols. Support for different communication protocols is provided by means of supplemental communication protocol specifications (e.g. for PROFINET, PROFIBUS, Ethernet IP, TCP, HART and FF) or by means of manufacturer-specific protocol integration.

0.3 Presentation of OPC Unified Architecture

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 a hierarchical namespace containing data items and to read, to write and to monitor these items for data changes. The OPC Classic specifications are based on Microsoft COM/DCOM technology for the communication between software components from different vendors. Therefore OPC Classic server and clients are restricted to Windows OS based automation systems.

OPC UA incorporates all features of OPC Class specifications 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 first version of OPC UA is defining an optimized binary TCP protocol for high performance intranet communication as well as a mapping to accepted internet standards like Web Services. The abstract communication model does not depend on a specific protocol mapping and allows the addition of 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, DataType, ReferenceType and View. Each NodeClass has a defined set of Attributes.

This specification makes use of two essential OPC UA NodeClasses: Objects and Variables.

Objects are used to represent components of a system. An Object is associated with 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. An example for Properties of Objects is the Revision Property of a DeviceType.

DataVariables represent the contents of an Object. DataVariables might have component DataVariables. This is typically used by Servers to expose individual elements of arrays and structures. This specification uses DataVariables to represent data like the process variables provided by a device.

0.4 Presentation of OPC UA Device Integration

The specification "OPC UA Device Integration" is an extension of the overall OPC Unified Architecture specification series and defines the information model associated with Devices. The model is intended to provide a unified view of Devices irrespective of the underlying Device protocols. FDT deals with physical or logical Devices and the information model of IEC 62541-100 therefore is used as base for the FDT information model.

The Devices information model specifies different ObjectTypes and procedures used to represent devices and related components like the communication infrastructure in an OPC UA Address Space. The main use cases are device configuration and diagnostic, but it allows a general and standardized way for any kind of application to access device related information. The following examples illustrate the concepts used in this specification. See UA Devices for the complete definition of the Devices information model.

Figure 1 shows an example for a temperature controller represented as Device Object. The component ParameterSet contains all Variables describing the Device. The component MethodSet contains all Methods provided by the Device. Both components are inherited from the TopologyElementType which is the root Object type of the Device Object type hierarchy. Objects of the type FunctionalGroupType are used to group the Parameters and Methods of the Device into logical groups. The FunctionalGroupType and the grouping concept are defined in UA Devices but the groups are device type specific i.e. the groups ProcessData and Configuration are defined by the TemperatureControllerType in this example.

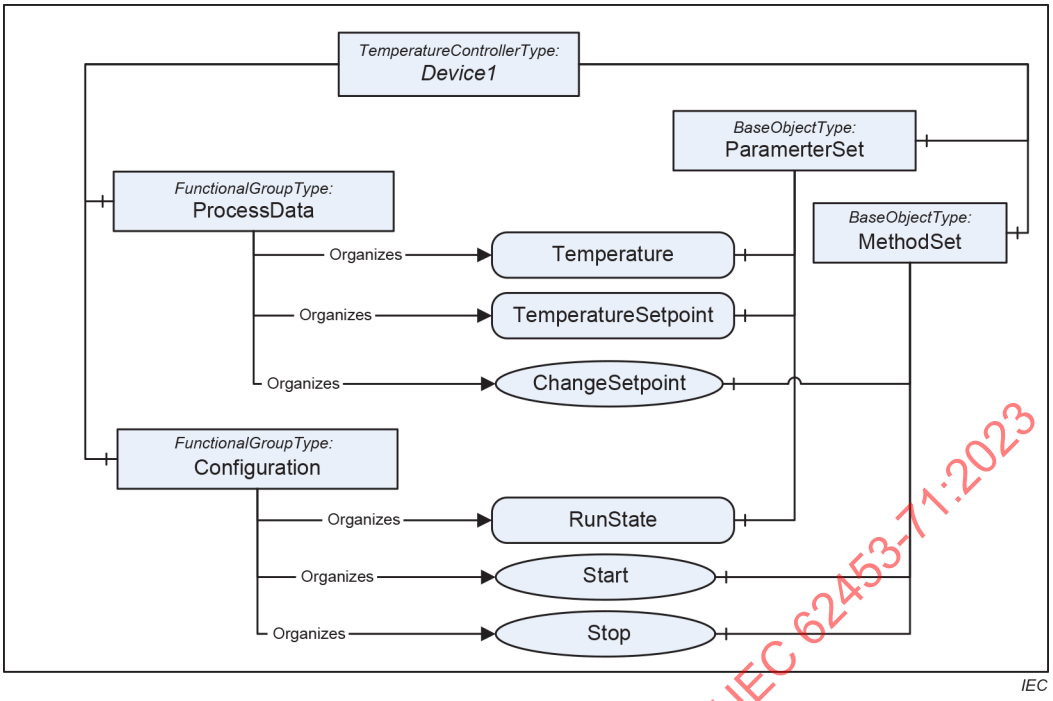


Figure 1 – OPC UA Devices Example

The use cases in Annex B illustrate the usage of the information model. Not all necessary Objects need to be realized within a concrete OPC UA Server.

FIELD DEVICE TOOL (FDT) INTERFACE SPECIFICATION –

Part 71: OPC UA Information Model for FDT

1 Scope

This part of IEC 62453 specifies an OPC UA Information Model to represent the device information based on FDT-defined device integration.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 62453-1:2023, *Field Device Tool (FDT) interface specification – Part 1: Overview and guidance*

IEC 62453-2:2022, *Field Device Tool (FDT) interface specification – Part 2: Concepts and detailed description*

IEC 62541-3:2020, *OPC Unified Architecture – Part 3: Address Space Model*

IEC 62541-5:2020, *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*

IEC 62541-100:2015, *OPC Unified Architecture – Part 100: Device Interface*

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 62453-1, IEC 62453-2 and IEC 62451-100 apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

- IEC Electropedia: available at https://www.electropedia.org/
- ISO Online browsing platform: available at https://www.iso.org/obp

3.2 Abbreviated terms

For the purposes of this document, the abbreviations given in IEC 62453-1, IEC 62453-2 as well as the following apply.

A&E	Alarms & Events
DA	Data Access
DTM	Device Type Manager
FDT	Field Device Technology
HDA	Historical Data Access
HMI	Human-Machine Interface
IEC	International Electrotechnical Commission
MES	Manufacturing Execution System
UA	Unified Architecture
XML	Extensible Markup Language

4 Conventions used in this document

4.1.1 Document conventions

Throughout this document certain document conventions are used.

Italics are used to denote a defined term or definition that appears in Clause 3 in this document or in one of the referenced OPC UA documents.

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.

4.1.2 Conventions for FDT methods

FDT defines synchronous methods and asynchronous methods. Asynchronous methods are implemented with a set of methods. For example an asynchronous method <MethodName> is implemented with Begin<MethodName>(), Cancel<MethodName>(), and End<MethodName>().

In this document asynchronous methods are indicated by '<>' in front of the name (e.g. <>MethodName).

4.1.3 Conventions for Node descriptions

4.1.3.1 General

Node definitions are specified using tables (see Table 2)

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

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

- If the *TargetNode* is a component of the *Node* being defined in the table the *Attributes* of the composed *Node* are defined in the same row of the table.
- The *DataType* is only specified for *Variables*; "[<number>]" indicates a single-dimensional array, for multi-dimensional arrays the expression is repeated for each dimension (e.g. [2][3] for a two-dimensional array). For all arrays the *ArrayDimensions* is set as identified by <number> values. If no <number> is set, the corresponding dimension is set to 0, indicating an unknown size. If no number is provided at all the

ArrayDimensions can be omitted. If no brackets are provided, it identifies a scalar *DataType* and the *ValueRank* is set to the corresponding value (see IEC 62541-3). In addition, *ArrayDimensions* is set to null or is omitted. If it can be Any or *ScalarOrOneDimension*, the value is put into "{<value>}", so either "{Any}" or "{*ScalarOrOneDimension*}" and the *ValueRank* is set to the corresponding value (see IEC 62541-3) and the *ArrayDimensions* is set to null or is omitted. Examples are given in Table 1.

Table 1 – Examples of DataTypes

Notation	**Data-Type**	**Value-Rank**	**Array-Dimensions**	**Description**
0:Int32	0:Int32	-1	omitted or null	A scalar Int32.
0:Int32[]	0:Int32	1	omitted or {0}	Single-dimensional array of Int32 with an unknown size.
0:Int32[][]	0:Int32	2	omitted or {0,0}	Two-dimensional array of Int32 with unknown sizes for both dimensions.
0:Int32[3][]	0: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.
0:Int32[5][3]	0: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.
0:Int32{Any}	0:Int32	-2	omitted or null	An Int32 where it is unknown if it is scalar or array with any number of dimensions.
0:Int32{ScalarOrOne Dimension}	0:Int32	-3	omitted or null	An Int32 where it is either a single-dimensional array or a scalar.

- The TypeDefinition is specified for *Objects* and *Variables*.
- The TypeDefinition column specifies a symbolic name for a *NodeId*, i.e. the specified *Node* points with a *HasTypeDefinition Reference* to the corresponding *Node*.
- The *ModellingRule* of the referenced component is provided by specifying the symbolic name of the rule in the *ModellingRule* column. In the *AddressSpace*, the *Node* shall use a *HasModellingRule Reference* to point to the corresponding *ModellingRule Object*.

If the *NodeId* of a *DataType* is provided, the symbolic name of the *Node* representing the *DataType* shall be used.

Note that if a symbolic name of a different namespace is used, it is prefixed by the *NamespaceIndex* (see 4.1.4.2).

Nodes of all other *NodeClasses* cannot be defined in the same table; therefore, only the used *ReferenceType*, their *NodeClass* and their *BrowseName* are specified. A reference to another part of this document points to their definition.

Table 2 illustrates the table. If no components are provided, the DataType, TypeDefinition and Other columns may be omitted and only a Comment column is introduced to point to the *Node* definition.

Table 2 – Example for type definition

Attribute	**Value**				
Attribute name	Attribute value. If it is an optional Attribute that is not set "--" is used.				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
ReferenceType name	NodeClass of the target Node.	BrowseName of the target Node.	DataType of the referenced Node, only applicable for Variables.	TypeDefinition of the referenced Node, only applicable for Variables and Objects.	Additional characteristics of the TargetNode such as the ModellingRule or AccessLevel.
NOTE Notes referencing footnotes of the table content.					

Components of *Nodes* can be complex that is containing components by themselves. The *TypeDefinition*, *NodeClass* and *DataType* can be derived from the type definitions, and the symbolic name can be created as defined in 4.1.5.1. Therefore, those containing components are not explicitly specified; they are implicitly specified by the type definitions.

The Other column defines additional characteristics of the Node. Examples of characteristics that can appear in this column are show in Table 3.

Table 3 – Examples of other characteristics

Name	**Short Name**	**Description**
0:Mandatory	M	The *Node* has the *Mandatory ModellingRule*.
0:Optional	O	The *Node* has the *Optional ModellingRule.*
0:MandatoryPlaceholder	MP	The *Node* has the *MandatoryPlaceholder ModellingRule.*
0:OptionalPlaceholder	OP	The *Node* has the *OptionalPlaceholder ModellingRule.*
ReadOnly	RO	The *Node AccessLevel* has the *CurrentRead* bit set but not the *CurrentWrite* bit.
ReadWrite	RW	The *Node AccessLevel* has the *CurrentRead* and *CurrentWrite* bits set.
WriteOnly	WO	The Node *AccessLevel* has the *CurrentWrite* bit set but not the *CurrentRead* bit.

If multiple characteristics are defined they are separated by commas. The name or the short name may be used.

4.1.3.2 Additional References

To provide information about additional *References*, the format as shown in Table 4 is used.

Table 4 – <some>Type Additional References

SourceBrowsePath	**Reference Type**	**Is Forward**	**TargetBrowsePath**
SourceBrowsePath is always relative to the *TypeDefinition*. Multiple elements are defined as separate rows of a nested table.	*ReferenceType* name	True = forward *Reference*.	TargetBrowsePath points to another *Node*, which can be a well-known instance or a *TypeDefinition*. You can use *BrowsePaths* here as well, which is either relative to the *TypeDefinition* or absolute. If absolute, the first entry needs to refer to a type or well-known instance, uniquely identified within a namespace by the *BrowseName*.

References can be to any other *Node*.

4.1.3.3 Additional sub-components

To provide information about sub-components, the format as shown in Table 5 is used.

Table 5 – <some>Type Additional sub-components

BrowsePath	References	NodeClass	BrowseName	DataType	TypeDefinition	Others
BrowsePath is always relative to the *TypeDefinition*. Multiple elements are defined as separate rows of a nested table	NOTE Same as for Table 2					

4.1.3.4 Additional Attribute values

The type definition table provides columns to specify the values for required *Node Attributes* for *InstanceDeclarations*. To provide information about additional *Attributes*, the format as shown in Table 6 is used.

Table 6 – <some>Type Attribute values for child Nodes

BrowsePath	<Attribute name> Attribute
BrowsePath is always relative to the *TypeDefinition*. Multiple elements are defined as separate rows of a nested table	The values of attributes are converted to text by adapting the reversible JSON encoding rules defined in IEC 52541-6. If the JSON encoding of a value is a JSON string or a JSON number then that value is entered in the value field. Double quotes are not included. If the DataType includes a NamespaceIndex (QualifiedNames, NodeIds or ExpandedNodeIds) then the notation used for BrowseNames is used. If the value is an Enumeration the name of the enumeration value is entered. If the value is a Structure then a sequence of name and value pairs is entered. Each pair is followed by a newline. The name is followed by a colon. The names are the names of the fields in the DataTypeDefinition. If the value is an array of non-structures then a sequence of values is entered where each value is followed by a newline. If the value is an array of Structures or a Structure with fields that are arrays or with nested Structures then the complete JSON array or JSON object is entered. Double quotes are not included.

There can be multiple columns to define more than one *Attribute*.

4.1.4 NodeIds and BrowseNames

4.1.4.1 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 *NamespaceUri* for all *NodeIds* defined in this document is defined in Annex A. The *NamespaceIndex* for this *NamespaceUri* is vendor-specific and depends on the position of the *NamespaceUri* in the server namespace table.

NOTE This document not only defines concrete *Nodes*, but also requires that some *Nodes* have to be generated, for example one for each *Session* running on the *Server*. The *NodeIds* of those *Nodes* are *Server*-specific, including the namespace. But the *NamespaceIndex* of those *Nodes* cannot be the *NamespaceIndex* used for the *Nodes* defined in this document, because they are not defined by this document but generated by the *Server*.

4.1.4.2 BrowseNames

The text part of the *BrowseNames* for all *Nodes* defined in this document is specified in the tables defining the *Nodes*. The *NamespaceUri* for all *BrowseNames* defined in this document is defined in 13.2.

For *InstanceDeclarations* of *NodeClass Object* and *Variable* that are placeholders (*OptionalPlaceholder* and *MandatoryPlaceholder ModellingRule*), the *BrowseName* and the *DisplayName* are enclosed in angle brackets (<>) as recommended in IEC 62541-3. If the *BrowseName* is not defined by this document, a namespace index prefix is added to the *BrowseName* (e.g., prefix '0' leading to '0:EngineeringUnits' or prefix '2' leading to '2:DeviceRevision'). This is typically necessary if a *Property* of another specification is overwritten or used in the OPC UA types defined in this document. Table 82 provides a list of namespaces and their indexes as used in this document.

4.1.5 Common Attributes

4.1.5.1 General

The *Attributes* of *Nodes*, their *DataTypes* and descriptions are defined in IEC 62541-3. Attributes not marked as optional are mandatory and shall be provided by a *Server*. The following tables define if the *Attribute* value is defined by this document or if it is server-specific.

For all *Nodes* specified in this document, the *Attributes* named in Table 7 shall be set as specified in the table.

Table 7 – Common Node Attributes

Attribute	Value
DisplayName	The *DisplayName* is a *LocalizedText*. Each *Server* shall provide the *DisplayName* identical to the *BrowseName* of the *Node* for the *LocaleId* "en". Whether the server provides translated names for other *LocaleIds* are server-specific.
Description	Optionally a server-specific description is provided.
NodeClass	Shall reflect the *NodeClass* of the *Node*.
NodeId	The *NodeId* is described by *BrowseNames* as defined in 4.1.4.1.
WriteMask	Optionally the *WriteMask Attribute* can be provided. If the *WriteMask Attribute* is provided, it shall set all non-server-specific *Attributes* to not writable. For example, the *Description Attribute* may be set to writable since a *Server* may provide a server-specific description for the *Node*. The *NodeId* shall not be writable, because it is defined for each *Node* in this document.
UserWriteMask	Optionally the *UserWriteMask Attribute* can be provided. The same rules as for the *WriteMask Attribute* apply.
RolePermissions	Optionally server-specific role permissions can be provided.
UserRolePermissions	Optionally the role permissions of the current Session can be provided. The value is server-specific and depends on the *RolePermissions Attribute* (if provided) and the current *Session*.
AccessRestrictions	Optionally server-specific access restrictions can be provided.

4.1.5.2 Objects

For all *Objects* specified in this document, the *Attributes* named in Table 8 shall be set as specified in the Table 8. The definitions for the *Attributes* can be found in IEC 62541-3.

Table 8 – Common Object Attributes

Attribute	Value
EventNotifier	Whether the *Node* can be used to subscribe to *Events* or not is server-specific.

4.1.5.3 Variables

For all *Variables* specified in this document, the *Attributes* named in Table 9 shall be set as specified in the table. The definitions for the *Attributes* can be found in IEC 62541-3.

Table 9 – Common Variable Attributes

Attribute	Value
MinimumSamplingInterval	Optionally, a server-specific minimum sampling interval is provided.
AccessLevel	The access level for *Variables* used for type definitions is server-specific, for all other *Variables* defined in this document, the access level shall allow reading; other settings are server-specific.
UserAccessLevel	The value for the *UserAccessLevel Attribute* is server-specific. It is assumed that all *Variables* can be accessed by at least one user.
Value	For *Variables* used as *InstanceDeclarations*, the value is server-specific; otherwise it shall represent the value described in the text.
ArrayDimensions	If the *ValueRank* does not identify an array of a specific dimension (i.e. *ValueRank* <= 0) the *ArrayDimensions* can either be set to null or the *Attribute* is missing. This behaviour is server-specific. If the *ValueRank* specifies an array of a specific dimension (i.e. *ValueRank* > 0) then the *ArrayDimensions Attribute* shall be specified in the table defining the *Variable*.
Historizing	The value for the *Historizing Attribute* is server-specific.
AccessLevelEx	If the *AccessLevelEx Attribute* is provided, it shall have the bits 8, 9, and 10 set to 0, meaning that read and write operations on an individual *Variable* are atomic, and arrays can be partly written.

4.1.5.4 VariableTypes

For all *VariableTypes* specified in this document, the *Attributes* named in Table 10 shall be set as specified in the table. The definitions for the *Attributes* can be found in IEC 62541-3.

Table 10 – Common VariableType Attributes

Attributes	Value
Value	Optionally a server-specific default value can be provided.
ArrayDimensions	If the *ValueRank* does not identify an array of a specific dimension (i.e. *ValueRank* <= 0) the *ArrayDimensions* can either be set to null or the *Attribute* is missing. This behaviour is server-specific. If the *ValueRank* specifies an array of a specific dimension (i.e. *ValueRank* > 0) then the *ArrayDimensions Attribute* shall be specified in the table defining the *VariableType*.

4.1.5.5 Methods

For all *Methods* specified in this document, the *Attributes* named in Table 11 shall be set as specified in the table. The definitions for the *Attributes* can be found in IEC 62541-3.

Table 11 – Common Method Attributes

Attributes	Value
Executable	All *Methods* defined in this document shall be executable (*Executable Attribute* set to "True"), unless it is defined differently in the *Method* definition.
UserExecutable	The value of the *UserExecutable Attribute* is server-specific. It is assumed that all *Methods* can be executed by at least one user.

4.1.6 Graphical notation

4.1.6.1 General

OPC UA defines a graphical notation for an OPC UA *AddressSpace*. It defines graphical symbols for all NodeClasses and how different types of *References* between Nodes can be visualized. Figure 2 shows the symbols for the NodeClasses used in this specification. NodeClasses representing types always have a shadow.

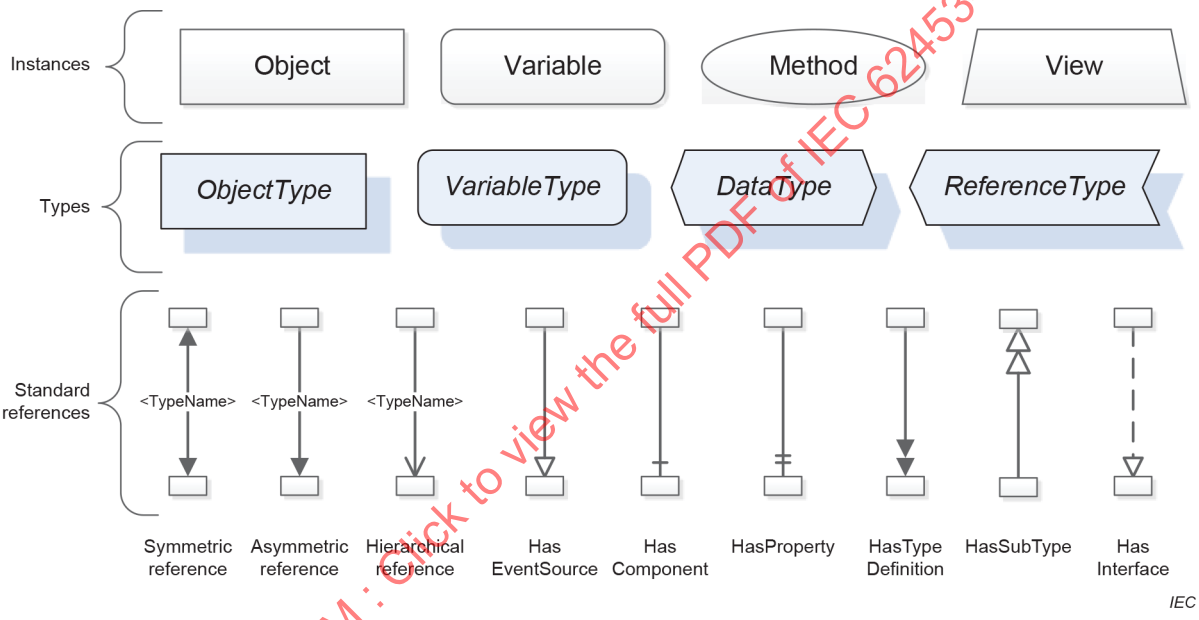


Figure 2 – The OPC UA Information Model Notation

A complete description of the different types of Nodes and References can be found in IEC 62541-3 and the base structure is described in IEC 62541-5.

OPC UA specification defines a very wide range of functionality in its basic information model. It is not required that all *Clients* or *Servers* support all functionality in the OPC UA specifications. OPC UA includes the concept of *Profiles*, which segment the functionality into testable certifiable units. This allows the definition of functional subsets (that are expected to be implemented) within a companion specification. The *Profiles* do not restrict functionality, but generate requirements for a minimum set of functionality (see IEC 62541-7).

4.1.6.2 Namespaces

OPC UA allows information from many different sources to be combined into a single coherent *AddressSpace*. Namespaces are used to make this possible by eliminating naming and id conflicts between information from different sources. Each namespace in OPC UA has a globally unique string called a NamespaceUri which identifies a naming authority and a locally unique integer called a *NamespaceIndex*, which is an index into the *Server's* table of *NamespaceUris*. The *NamespaceIndex* is unique only within the context of a *Session* between an OPC UA *Client* and an OPC UA *Server*- the *NamespaceIndex* can change between *Sessions* and still identify the same item even though the NamespaceUri's location in the table has changed. The *Services* defined for OPC UA use the *NamespaceIndex* to specify the Namespace for qualified values.

There are two types of structured values in OPC UA that are qualified with *NamespaceIndexes*: NodeIds and *QualifiedNames*. NodeIds are locally unique (and sometimes globally unique) identifiers for *Nodes*. The same globally unique *NodeId* can be used as the identifier in a node in many *Servers* – the node's instance data may vary but its semantic meaning is the same regardless of the *Server* it appears in. This means *Clients* can have built-in knowledge of of what the data means in these *Nodes*. OPC UA *Information Models* generally define globally unique *NodeIds* for the *TypeDefinitions* defined by the *Information Model*.

QualifiedNames are non-localized names qualified with a Namespace. They are used for the *BrowseNames* of *Nodes* and allow the same names to be used by different information models without conflict. *TypeDefinitions* are not allowed to have children with duplicate *BrowseNames*; however, instances do not have that restriction.

5 Concept

5.1 System architecture

The system architecture as shown in Figure 3 has already been defined in IEC 62453-42:2016. It shall follow the general FDT software architecture as defined in IEC 62453-1 and may be applied to all object implementation profiles based on IEC 62453-2.

The OPC UA server is a Frame Application. The Frame Application is interacting with the DTMs to manage the device-specific information and in order to interact with the devices. Frame Applications may support different use cases, for example: vendor-specific service, process control or asset management. Frame Applications may start and terminate the DTMs on demand (e.g. only if a value is read from the respective device).

The OPC UA server is equipped with an OPC UA server interface. In the information model of the OPC UA server the application-specific project information may be represented (depending on the Frame Application) and device-specific information shall be represented. The OPC UA client may browse the information model and may use the services of the OPC UA interface to access the represented information. If device-specific information is accessed, the information is retrieved from the DTMs.

In a multi-client scenario, the Frame Application is responsible for managing the access to the DTMs. The approach for managing the access from multiple clients is to use the "multi-user access" as specified in IEC 62453-42.

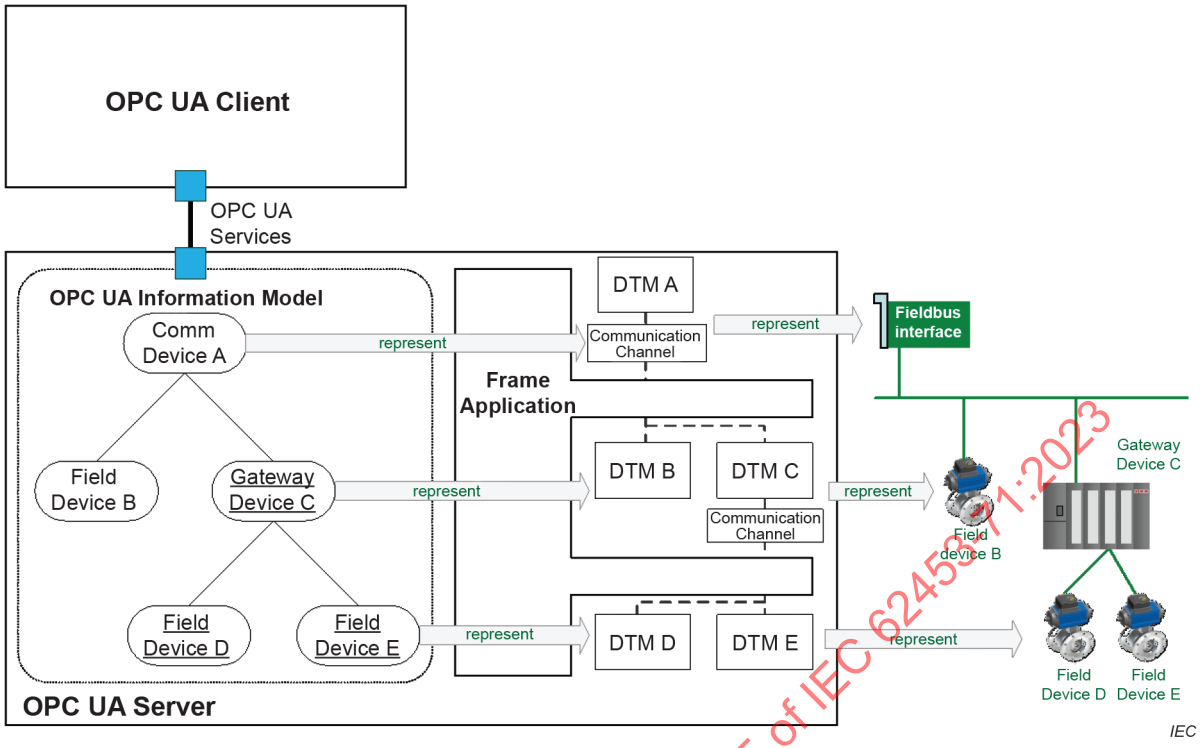


Figure 3 – System architecture according to IEC 62453-42

Since the contents of the FDT project and the management of multi-client access is specific to the Frame Application, this document focuses on the mapping of the information from the DTM to the 'Device model' of OPC UA DI. One possible approach for presentation of the project information would be the 'Device Integration Host Model' according to OPC UA DI.

6 FDT specific OPC UA ObjectTypes

6.1 General

The OPC UA information model for FDT is based on IEC 62541-100. This document defines a mapping for device-related information, services and data types.

Services which are defined in IEC 62541-100, but for which no mapping is defined here (e.g. Locking service), have to be implemented by the OPC UA Server according to the specification in IEC 62541-100.

6.2 FdtDeviceType

This OPC UA *ObjectType* is the base for a device type derived from an FDT device type definition. Any manufacturer-specific device type is derived from *FdtDeviceType*.

Figure 4 shows an overview for the *FdtDeviceType* with its Properties and related ObjectTypes. It is formally defined in Table 12.

NOTE Specific DeviceType is an example depicting how device types for specific devices inherit from the *FdtDeviceType*.

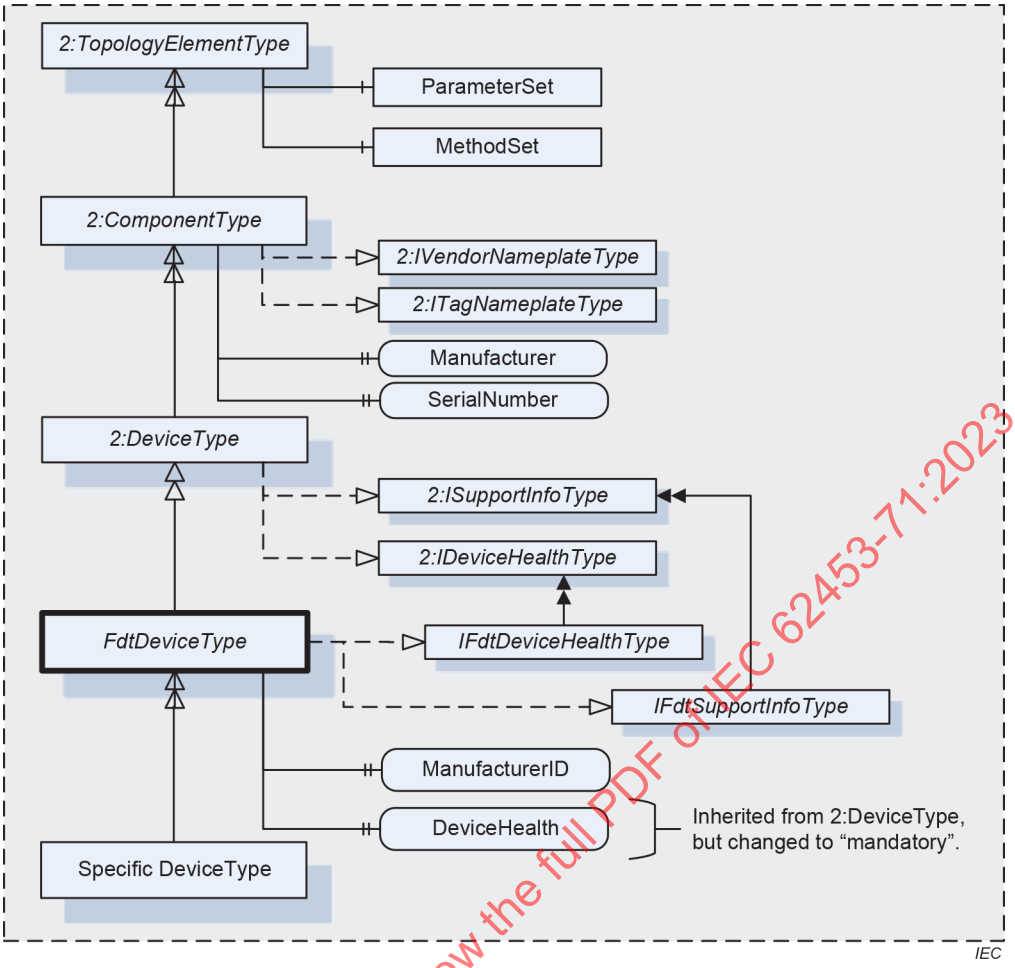


Figure 4 – FdtDeviceType overview

The *FdtDeviceType* is formally defined in Table 12.

Table 12 – FdtDeviceType definition

Attribute	**Value**				
BrowseName	FdtDeviceType				
IsAbstract	True				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 2:DeviceType defined in IEC 62541-100					
HasInterface	ObjectType	IFdtDeviceHealthType		Defined in 6.4	
HasInterface	ObjectType	IFdtSupportInfoType		Defined in 6.5	
0:HasProperty	Variable	ManufacturerId	0:String	0:PropertyType	O
0:HasProperty	Variable	DeviceTypeId	0:String	0:PropertyType	O
0:HasProperty	Variable	DeviceTag	0:String	0:PropertyType	M
Applied from 2:IFdtDeviceHealthType					
0:HasComponent	Variable	2:DeviceHealth	2:DeviceHealth Enumeration	0:BaseDataVariable Type	M, RO[a)]
[a)] This definition for *DeviceHealth* overrides the definition of [1][1].					

[1] Numbers in square brackets refer to the bibliography.

The *FdtDeviceType* is an abstract type and cannot be used directly.

The mapping of FDT information to FdtDeviceType is defined in 11.2.1.

defined in

6.3 FdtFunctionalGroupType

FdtFunctionalGroupType is used for representing functional grouping of methods and parameters. It is formally defined in Table 13.

Table 13 – FdtFunctionalGroupType definition

Attribute	**Value**				
BrowseName	FdtFunctionalGroupType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 2:FunctionalGroupType defined in IEC 62541-100:2015, 5.3					
0:HasProperty	Variable	ApplicationId	ApplicationIdEnumeration	0:PropertyType	O
0:HasProperty	Variable	SemanticInfo	SemanticInfoType	0:PropertyType	O

The *FdtFunctionalGroupType* is a concrete type and can be used directly.

6.4 IFdtDeviceHealthType interface

IFdtDeviceHealthType defines additional requirements for the *IDeviceHealthType* interface, which is formally defined in Table 14.

Table 14 – IFdtDeviceHealthType definition

Attribute	**Value**				
BrowseName	IFdtDeviceHealthType				
IsAbstract	True				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 2:IDeviceHealthType interface[a].					
0:HasComponent	Variable	2:DeviceHealth	2:DeviceHealthEnumeration	0:BaseDataVariableType	M[b]

[a] IDeviceHealthType is explained in [1], 5.5.4.

[b] The definition for *DeviceHealth* overrides the definition of [1].

6.5 IFdtSupportInfoType interface

6.5.1 Overview

The *IFdtSupportInfoType* defines additional requirements for the *ISupportInfoType* interface, which is formally defined in Table 15.

Table 15 – IFdtSupportInfoType definition

Attribute	Value				
BrowseName	IFdtSupportInfoType				
IsAbstract	True				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 2:ISupportInfoType[a].					
[a] ISupportInfoType is explained in [1], 5.5.5.					

The components of the *IFdtSupportInfoType* have additional references as defined in Table 16.

Table 16 – IFdtSupportInfoType additional subcomponents

Source Path	References	Node Class	BrowseName	Data Type	TypeDefinition	Other
2:Documentation	HasComponent	Object	<FdtDocumentIdentifier>		FdtDocumentType	MP
2:ProtocolSupport	HasComponent	Object	<FdtProtocolSupportIdentifier>		FdtDocumentType	MP

Documents provided for a device are exposed as *Objects* organized in the *Documentation* folder. In most cases they will represent a product manual, which can exist as a set of individual documents. The *BrowseName* of each *Object* in the *Documentation* Folder will consist of the document label that can be used to identify the document.

Protocol support files (e.g. GSD files, EDS files) provided for a device are exposed as *Objects* organised in the *ProtocolSupport* folder.

6.6 Document types

6.6.1 FdtDocumentType

This abstract type describes the information associated with a document provided by a DTM. The *FdtDocumentType* is formally described in Table 17.

Table 17 – FdtDocumentType definition

Attribute	Value				
BrowseName	FdtDocumentType				
IsAbstract	True				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 0:BaseObjectType					
0:HasProperty	Variable	Classification	DocumentClassification	0:PropertyType	M
0:HasProperty	Variable	Help	0:String	0:PropertyType	O
0:HasProperty	Variable	Language	0:String	0:PropertyType	O
0:HasProperty	Variable	MediaType	0:String	0:PropertyType	M
0:HasProperty	Variable	SemanticInfo	SemanticInfoType	0:PropertyType	O

The *FdtDocumentType* is an abstract type and cannot be used directly.

The MIME type of document is identified by the *MediaType* property of each *Variable*.

6.6.2 FdtDocumentFile

This is a type for representing documents, which are provided as files. It is formally defined in Table 18.

Table 18 – FdtDocumentFile definition

Attribute	Value				
BrowseName	FdtDocumentFile				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of FdtDocumentType					
0:HasComponent	Object	File		0:FileType	M

The *FdtDocumentFile* is a concrete type and can be used directly.

The document may be retrieved from the server by using the methods related to the object of type *FileType*.

6.6.3 FdtDocumentUrl

This is a type for representing documents, which are provided as URL. It is formally defined in Table 19.

Table 19 – FdtDocumentUrl definition

Attribute	Value				
BrowseName	FdtDocumentUrl				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of FdtDocumentType					
0:HasProperty	Variable	URL	0:String	0:PropertyType	M

The *FdtDocumentUrl* is a concrete type and can be used directly.

The actual document may be retrieved by the *Client* from the URL.

6.7 FdtProtocolType

This type is used to specify which protocols an *FdtDeviceType* requires or supports (see Figure 5).

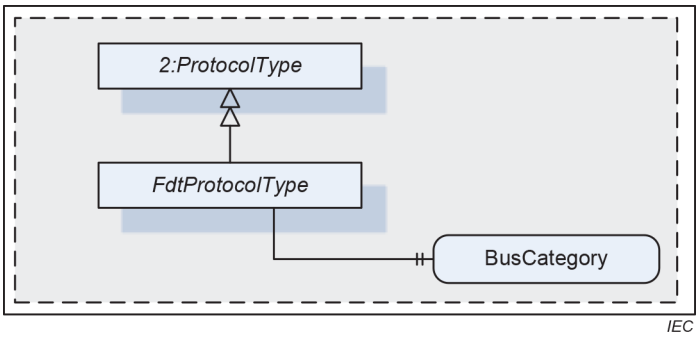


Figure 5 – FdtProtocolType overview

The *FdtProtocolType* is formally defined in Table 20.

Table 20 – FdtProtocolType definition

Attribute	Value				
BrowseName	FdtProtocolType				
IsAbstract	True				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 2:ProtocolType defined in IEC 62541-100					
0:HasProperty	Variable	BusCategory	0:String	0:PropertyType	M

The *FdtProtocolType* is an abstract type and cannot be used directly.

The property *BusCategory* shall be used by any instance of *FdtProtocolType* or its subtypes.

6.8 FdtTransferServiceType

This type is used to specify support for transfer services (see Figure 6).

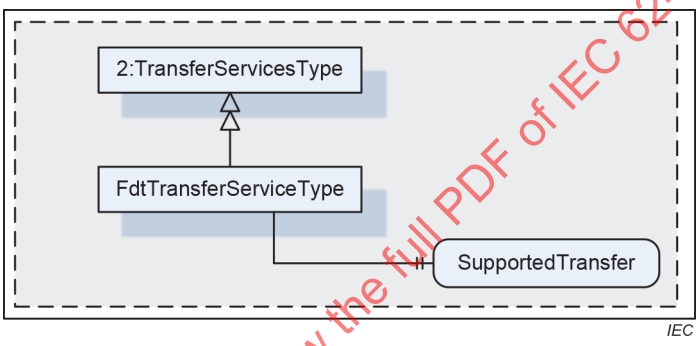


Figure 6 – FdtTransferServiceType overview

The *FdtTransferServiceType* is formally defined in Table 21.

Table 21 – FdtTransferServiceType definition

Attribute	Value				
BrowseName	FdtTransferServiceType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of 2:TransferServicesType					
0:HasProperty	Variable	SupportedTransfer	SupportedTransfer	0:PropertyType	M

The *FdtTransferServiceType* is a concrete type and can be used directly.

The property *SupportedTransfer* (defined in 9.4.11) is used to indicate the services supported for the respective device.

6.9 FdtIoSignalInfoType

This type is used to provide information about the IO signals available at the device (see Figure 7).

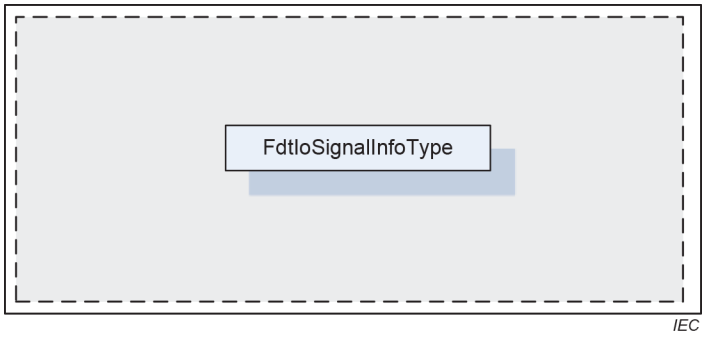


Figure 7 – FdtIoSignalInfoType overview

The *FdtIoSignalInfoType* is formally defined in Table 22.

Table 22 – FdtIoSignalInfoType definition

Attribute	**Value**				
BrowseName	FdtIoSignalInfoType				
IsAbstract	False				
References	**Node Class**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of BaseObjectType					
0:HasProperty	Variable	Description	0:String	0:PropertyType	O
0:HasProperty	Variable	FrameApplicationTag	0:String	0:PropertyType	M
0:HasProperty	Variable	IECDatatype	IECDatatype	0:PropertyType	M
0:HasProperty	Variable	IsLocked	0:Boolean	0:PropertyType	M
0:HasProperty	Variable	IsSafety	0:Boolean	0:PropertyType	M
0:HasProperty	Variable	RoutedIoSignalId	0:String	0:PropertyType	O
0:HasProperty	Variable	SemanticInfo	SemanticInfoType	0: PropertyType	O
0:HasProperty	Variable	SignalType	SignalTypeEnumeration	0:PropertyType	M
0:HasProperty	Variable	HasAlarmInfo	DataRefType	0:PropertyType	O
0:HasProperty	Variable	HasDeviceData	DataRefType	0:PropertyType	O
0:HasProperty	Variable	HasRange	DataRefType	0:PropertyType	O
0:HasProperty	Variable	HasSubstituteValue	DataRefType	0:PropertyType	O
0:HasProperty	Variable	HasUnit	DataRefType	0:PropertyType	O

The *FdtIoSignalInfoType* is a concrete type and can be used directly.

7 OPC UA EventTypes

7.1 Overview

In order to support audit trail, new audit trail event types are defined (see Figure 8).

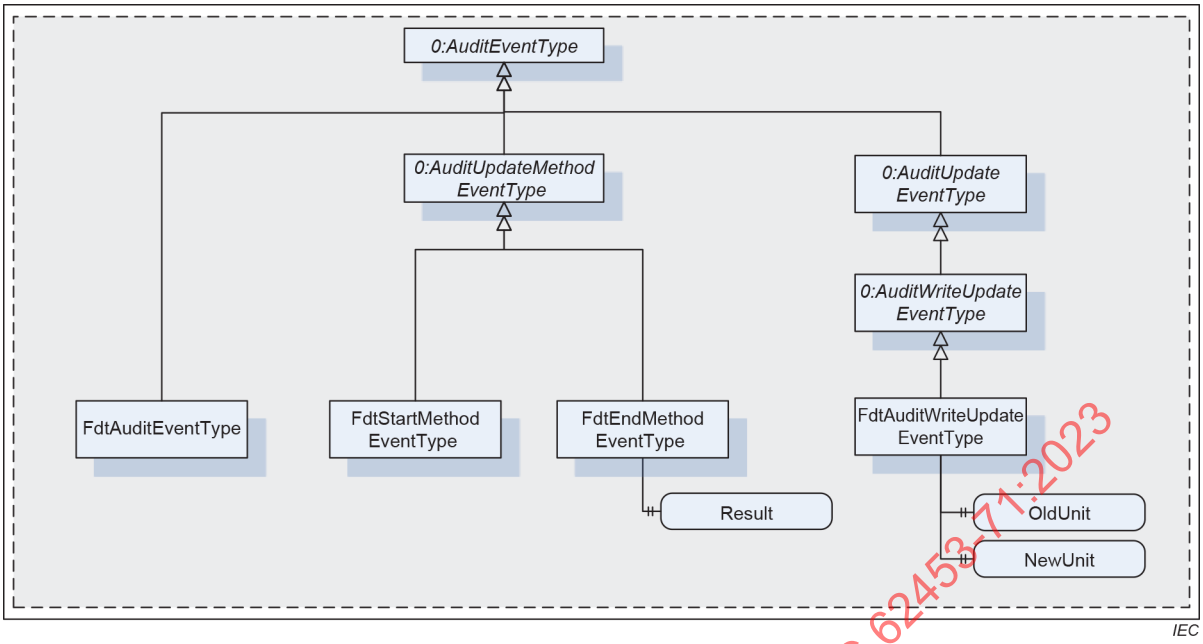


Figure 8 – Audit event type overview

7.2 FdtAuditEventType

The *FdtAuditEventType* is an event type for general events (e.g. for device status update). It is formally defined in Table 23.

Table 23 – FdtAuditEventType definition

Attribute		Value			
BrowseName		FdtAuditEventType			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:AuditEventType defined in IEC 62541-3 Address Space Model 9.5, which means it inherits the InstanceDeclarations of that Node.					

7.3 FdtStartMethodEventType

The *FdtStartMethodEventType* is an event type for indication of start of execution of a command function. It is formally defined in Table 24.

Table 24 – FdtStartMethodEventType definition

Attribute		Value			
BrowseName		FdtStartMethodEventType			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:AuditUpdateMethodEventType defined in 7.2, which means it inherits the InstanceDeclarations of that Node.					

7.4 FdtEndMethodEventType

The *FdtEndMethodEventType* is an event type for indication of end of execution of a command function. It is formally defined in Table 25.

Table 25 – FdtEndMethodEventType definition

Attribute		Value			
BrowseName		FdtEndMethodEventType			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:AuditUpdateMethodEventType defined in 7.2, which means it inherits the InstanceDeclarations of that Node.					
0:HasProperty	Variable	Result	FunctionExecutionResultState	0:PropertyType	M

The property Result is used to indicate the result of the function execution.

7.5 FdtAuditWriteUpdateEventType

The *FdtAuditWriteUpdateEventType* is used to indicate the change of a parameter value. It is formally defined in Table 26.

Table 26 – FdtAuditWriteUpdateEventType definition

Attribute		Value			
BrowseName		FdtAuditWriteUpdateEventType			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:AuditWriteUpdateEventType defined in IEC 62541-3 Address Space Model 9.27, which means it inherits the InstanceDeclarations of that Node.					
0:HasProperty	Variable	OldUnit	0:String	0:PropertyType	O
0:HasProperty	Variable	NewUnit	0:String	0:PropertyType	O

The property *OldUnit* may be used to indicate the unit of the old value. The property *NewUnit* may be used to indicate the unit of the new value. If the source node of the event has an engineering unit, it is mandatory to provide these properties.

8 OPC UA VariableTypes

8.1 FdtParameter

A specific type *FdtParameter* is defined, which is an extension of *DataItemType* (see Table 27).

Table 27 – FdtParameter definition

Attribute		Value			
BrowseName		FdtParameter			
IsAbstract		False			
ValueRank		−2 (−2 = Any)			
DataType		0:BaseDataType			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of DataItemType/ HasTypeDefinition FdtParameter					
0:HasProperty	Variable	DisplayFormat	0:String	0:PropertyType	O
0:HasProperty	Variable	AlarmType	AlarmType	0:PropertyType	O
0:HasProperty	Variable	RangeType	RangeType	0:PropertyType	O
0:HasProperty	Variable	SubstitutionType	SubstitutionType	0:PropertyType	O
0:HasProperty	Variable	ApplicationId	ApplicationIdEnumeration	0:PropertyType	O
0:HasProperty	Variable	0:EURange	0:Range	0:PropertyType	O
0:HasProperty	Variable	0:EngineeringUnits	0:EUInformation	0:PropertyType	O
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText[]	0:PropertyType	O
0:HasProperty	Variable	SemanticInfo	SemanticInfoType	0:PropertyType	O
0:HasProperty	Variable	DataRef	DataRefType	0:PropertyType	O
0:HasProperty	Variable	AlarmDataRef	DataRefType	0:PropertyType	O
0:HasProperty	Variable	RangeDataRef	DataRefType	0:PropertyType	O
0:HasProperty	Variable	SubstituteDataRef	DataRefType	0:PropertyType	O
NonHierarchicalReferences					
HasIOSignalRef	ObjectType	FdtIoSignalInfoType			

The *FdtParameter* is a concrete type and can be used directly.

DisplayFormat is a format string for numerical data, compliant to .NET string format definitions.

If the *DataType* of the FdtParameter is an *Enumeration DataType*, then *EnumStrings* are used according to IEC 62541-3.

EURange and *EngineeringUnits* are used as defined in IEC 62541-8 for the *AnalogueItem* variable type.

SemanticInfo provides additional semantic information for the parameter (e.g., a reference to a definition in a device profile).

If the *FdtParameter* represents a structured data information, then *ApplicationId* may be used to categorize the use case for the data structure.

The *DataRefType* properties are used to reference related *FdtParameters*:

– DataRef, provides references to other data. Information about the type of reference is provided by the SemanticInfo of the *DataRefType*.
– AlarmDataRef, provides references to alarm information,
– RangeDataRef, provides references to range information, and
– SubstituteDataRef, provides references to substitute value settings.

If the *FdtParameter* represents alarm information, then *AlarmType* categorizes the available alarm type.

If the *FdtParameter* represents range information, then *RangeType* distinguishes whether the data represents an upper range or a lower range.

If the *FdtParameter* represents substitute information, then *SubstitutionType* specifies which value shall be used when a substitute value is needed.

The *HasIOSignalRef* reference, provides a reference to the description of an IO signal that corresponds to the *FdtParameter*.

9 OPC UA DataTypes

9.1 DataRefType

The *DataRefType* provides a reference to a data item. It is formally defined in Table 28.

Table 28 – DataRefType structure

Name	Type	Description
DataRefType	structure	Reference to a data item.
DataId	0:String	Id of the referenced data item.
SemanticInfo	SemanticInfoType	Meaning of the data item in the current context.

The representation of *DataRefType* in the *AddressSpace* is defined in Table 29.

Table 29 – DataRefType definition

Attribute	Value				
BrowseName	DataRefType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:Structure defined in IEC 62541-3.					

9.2 FdtDeviceClassificationType

The *FdtDeviceClassificationType* is used to classify devices. It is formally defined in Table 30.

Table 30 – FdtDeviceClassificationType structure

Name	Type	Description
FdtDeviceClassificationType	structure	Classification of a device according to IEC 62390 Annex G.
ClassificationDomain	ClassificationDomainId	Device classification domain groups.
DeviceClassification	ClassificationId	Unique identifiers according to device primary function.

The representation of *FdtDeviceClassificationType* in the *AddressSpace* is defined in Table 31.

Table 31 – FdtDeviceClassificationType definition

Attribute		Value			
BrowseName		FdtDeviceClassificationType			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:Structure defined in IEC 62541-3.					

9.3 SemanticInfoType

This type describes semantic information associated with data provided by a DTM. The *SemanticInfoType* is formally defined in Table 32.

Table 32 – SemanticInfoType structure

Name	Type	Description
SemanticInfoType	structure	Semantic information associated with data provided by a DTM.
ApplicationDomain	0:String	Application domain.
SemanticId	0:String	Semantic identifiers in domain.

The representation of *SemanticInfoType* in the *AddressSpace* is defined in Table 33.

Table 33 – SemanticInfoType definition

Attribute		Value			
BrowseName		SemanticInfoType			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the 0:Structure defined in IEC 62541-3.					

9.4 Enumeration datatypes

9.4.1 AlarmType

AlarmType is an enumeration that defines the available alarm types. The values shall be mapped as defined in Table 34.

Table 34 – AlarmType items

Name	Value	Description
HighHighAlarm	0	Alarm if a process value exceeds ‘highhigh’ limit.
HighAlarm	1	Alarm if a process value exceeds ‘high’ limit.
LowLowAlarm	2	Alarm if a process value falls below ‘lowlow’ limit.
LowAlarm	3	Alarm if a process value falls below ‘low’ limit.

Its representation in the *AddressSpace* is defined in Table 35.

Table 35 – AlarmType definition

Attribute		Value			
BrowseName		AlarmType			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in IEC 62541-5					
0:HasProperty	Variable	0:EnumValues	0: EnumValueType[]	0:PropertyType	

9.4.2 ApplicationIdEnumeration

The *ApplicationIdEnumeration* is an enumeration that defines the use of the *FunctionalGroup*. Its values are defined in Table 36.

Table 36 – ApplicationIdEnumeration items

Name	Value	Description
AdjustSetValue	0	Functional group is used to adjust the set value.
AssetManagement	1	Functional group is used for asset management.
AuditTrail	2	Functional group is used for audit trail.
Configuration	3	Functional group is used for configuration.
Diagnosis	4	Functional group is used for diagnosis.
Force	5	Functional group is used for forcing values.
Observe	6	Functional group is used for observation of the device.
OfflineCompare	7	Functional group is used for comparison of offline data from different devices.
OfflineParameterize	8	Functional group is used for offline parameterization.
OnlineCompare	9	Functional group is used for comparison of the device dataset and the instance dataset.
OnlineParameterize	10	Functional group is used for online parameterization.
Identify	11	Functional group is used for identification.
Calibration	12	Functional group is used for calibration.
MainOperation	13	Functional group is the aggregation of all functional groups.
NetworkManagement	14	Functional group is used for network management.

Its representation in the *AddressSpace* is defined in Table 37.

Table 37 – ApplicationIdEnumeration definition

Attribute		Value			
BrowseName		ApplicationIdEnumeration			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in IEC 62541-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType[]	0:PropertyType	

9.4.3 ClassificationDomainId

ClassificationDomainId is an enumeration that defines the available domains for device classifications. Its values are defined in Table 38.

Table 38 – ClassificationDomainId items

Name	Value	Description
PowerDistribution	0	Classification domain is PowerDistribution.
MotionControl	1	Classification domain is MotionControl.
Measurement	2	Classification domain is Measurement.
OperatorInterface	3	Classification is OperatorInterface.
ModulesAndControllers	4	Classification domain is ModulesAndControllers.
Communication	5	Classification domain is Communication.

Its representation in the *AddressSpace* is defined in Table 39.

Table 39 – ClassificationDomainId definition

Attribute	Value				
BrowseName	ClassificationDomainId				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in IEC 62541-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType []	0:PropertyType	

9.4.4 ClassificationId

ClassificationId is an enumeration that defines the available classifications for devices. Its values are defined in Table 40.

Table 40 – ClassificationId items

Name	Value	Description
Flow	0	Classification is Flow.
Level	1	Classification is Level.
Pressure	2	Classification is Pressure.
Temperature	3	Classification is Temperature.
Valve	4	Classification is Valve.
Positioner	5	Classification is Positioner.
Actuator	6	Classification is Actuator.
Nc_rc	7	Classification is Numeric Control / Robotic Control.
Encoder	8	Classification is Encoder.
SpeedDrive	9	Classification is SpeedDrive.
Hmi	10	Classification is HMI.
AnalogInput	11	Classification is AnalogInput.
AnalogOutput	12	Classification is AnalogOutput.
DigitalInput	13	Classification is DigitalInput.
DigitalOutput	14	Classification is DigitalOutput.
ElectrochemicalAnalyser	15	Classification is ElectrochemicalAnalyser.
DtmSpecific	16	Classification is DtmSpecific.
Universal	17	Classification is Universal.
Analyser	18	Classification is Analyser.
RemoteIO	19	Classification is RemoteIO.
AnalogCombinedIO	20	Classification is AnalogCombinedIO.
DigitalCombinedIO	21	Classification is DigitalCombinedIO.
Recorder	22	Classification is Recorder.

Name	Value	Description
Controller	23	Classification is Controller.
Angle	24	Classification is Angle.
LimitSwitch	25	Classification is LimitSwitch.
Converter	26	Classification is Converter.
Motor	27	Classification is Motor.
Switchboard	28	Classification is Switchboard.
CircuitBreaker	29	Classification is CircuitBreaker.
PowerMonitoring	30	Classification is PowerMonitoring.
DistributionPanel	31	Classification is DistributionPanel.
Contactor	32	Classification is Contactor.
ProtectionStarter	33	Classification is ProtectionStarter.
SoftStarter	34	Classification is SoftStarter.
Drive	35	Classification is Drive.
AxisControl	36	Classification is AxisControl.
MotorControlCenter	37	Classification is MotorControlCenter.
ControlValve	38	Classification is ControlValve.
Electrical	39	Classification is Electrical.
Density	40	Classification is Density.
Quality	41	Classification is Quality.
SpeedOrRotaryFrequency	42	Classification is SpeedOrRotaryFrequency.
Radiation	43	Classification is Radiation.
WeightMass	44	Classification is WeightMass.
DistanceOrPositionPresence	45	Classification is DistanceOrPositionPresence.
PushButton	46	Classification is PushButton.
Joystick	47	Classification is Joystick.
Keypad	48	Classification is Keypad.
PilotLight	49	Classification is PilotLight.
StackLight	50	Classification is StackLight.
Display	51	Classification is Display.
CombinedButtonsAndLights	52	Classification is CombinedButtonsAndLights.
OperatorStation	53	Classification is OperatorStation.
GeneralInput	54	Classification is GeneralInput.
GeneralOutput	55	Classification is GeneralOutput.
CombinedInputOutput	56	Classification is CombinedInputOutput.
Relay	57	Classification is Relay.
Timer	58	Classification is Timer.
Scanner	59	Classification is Scanner.
ProgrammableController	60	Classification is ProgrammableController.
CommunicationAdapter	61	Classification is CommunicationAdapter.
Gateway	62	Classification is Gateway.

Its representation in the *AddressSpace* is defined in Table 41.

Table 41 – ClassificationId definition

Attribute		Value			
BrowseName		ClassificationId			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in IEC 62541-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType[]	0:PropertyType	

9.4.5 DocumentClassification

DocumentClassification is an enumeration that defines the available classes of documents. Its values are defined in Table 42.

Table 42 – DocumentClassification items

Name	Value	Description
Help	0	Document contains help information.
TechnicalDocumentation	1	Document contains technical information.
OrderingInformation	2	Document contains order information.
Miscellaneous	3	Document contains general information.
TenderSpecification	4	Document contains tender specification.

Its representation in the *AddressSpace* is defined in Table 43.

Table 43 – DocumentClassification definition

Attribute		Value			
BrowseName		DocumentClassification			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in IEC 62541-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType[]	0:PropertyType	

9.4.6 FunctionExecutionResultState

FunctionExecutionResultState is an enumeration that defines the type of result states for execution of a function provided for a device. Its values are defined in Table 44.

Table 44 – FunctionExecutionResultState items

Name	Value	Description
Cancel	0	The function was canceled.
Success	1	The function finished execution successfully.
Fail	2	The function execution failed.

Its representation in the *AddressSpace* is defined in Table 45.

Table 45 – FunctionExecutionResultState definition

Attribute		Value			
BrowseName		FunctionExecutionResultState			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in IEC 62541-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType[]	0:PropertyType	

9.4.7 IECDatatype

IECDatatype is an enumeration that defines the IEC type of an IOSignal. Its values are defined in Table 46.

Table 46 – IECDatatype items

Name	Value	Description
BOOL	0	1 bit -> true or false
SINT	1	Signed short integer (1byte)
INT	2	Signed integer (2 byte)
DINT	3	Signed double integer (4 byte)
LINT	4	Signed long integer (8 byte)
USINT	5	Unsigned short integer (1 byte)
UINT	6	Unsigned integer (2 byte)
UDINT	7	Unsigned double integer (4 byte)
ULINT	8	Unsigned long integer (8 byte)
REAL	9	Floating point (4 byte)
LREAL	10	Long floating point (8 byte)
TIME	11	Time
DATE	12	Calendar date
TimeOfDay	13	Clock time
DateAndTime	14	Date and time
STRING	15	Variable-length single-byte character string
BYTE	16	8 bit
WORD	17	16 bit
DWORD	18	32 bit
LWORD	19	64 bit
WSTRING	20	Variable-length double-byte character string

Its representation in the *AddressSpace* is defined in Table 47.

Table 47 – IECDatatype definition

Attribute		Value			
BrowseName		IECDatatype			
IsAbstract		False			
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in IEC 62541-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType[]	0:PropertyType	

9.4.8 RangeType

RangeType is an enumeration that defines the available types of range limits. Its values are defined in Table 48.

Table 48 – RangeType items

Name	Value	Description
LowerRange	0	Device data represents a lower range.
UpperRange	1	Device data represents an upper range.

Its representation in the *AddressSpace* is defined in Table 49.

Table 49 – RangeType definition

Attribute	Value				
BrowseName	RangeType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in IEC 62541-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType[]	0:PropertyType	

9.4.9 SignalTypeEnumeration

SignalTypeEnumeration is an enumeration that defines the type of an IO signal. Its values are defined in Table 50.

Table 50 – SignalTypeEnumeration items

Name	Value	Description
Input	0	Input signal
Output	1	Output signal

Its representation in the *AddressSpace* is defined in Table 51.

Table 51 – SignalTypeEnumeration definition

Attribute	Value				
BrowseName	SignalTypeEnumeration				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in IEC 62541-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType[]	0:PropertyType	

9.4.10 SubstitutionType

SubstitutionType is an enumeration that defines the type of substitution data. Its values are defined in Table 52.

Table 52 – SubstitutionType items

Name	Value	Description
LastValue	0	The last known value shall be used.
LastValidValue	1	The last valid value shall be used.
UpperRange	2	The upper range shall be used.
LowerRange	3	The lower range shall be used.

Its representation in the *AddressSpace* is defined in Table 53.

Table 53 – SubstitutionType definition

Attribute	Value				
BrowseName	SubstitutionType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in IEC 62541-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType[]	0:PropertyType	

9.4.11 SupportedTransfer

SupportedTransfer is an enumeration that defines the types of transfers provided for a device. Its values are defined in Table 54.

Table 54 – SupportedTransfer items

Name	Value	Description
None	0	The DTM does not support upload / download at all.
OnlyDownload	1	The DTM supports only writing data to the device. Reading data to the device is not supported.
OnlyUpload	2	The DTM supports only reading data from the device. Writing data to the device is not supported.
BothUploadAndDownload	3	The DTM supports both reading values from the device and writing values to the device.

Its representation in the *AddressSpace* is defined in Table 55.

Table 55 – SupportedTransfer definition

Attribute	Value				
BrowseName	SupportedTransfer				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Other**
Subtype of the Enumeration type defined in IEC 62541-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType[]	0:PropertyType	

10 OPC UA ReferenceTypes – HasIOSignalRef

The *HasIOSignalRef* ReferenceType is a concrete *ReferenceType* that can be used directly. It is a subtype of the *NonHierarchicalReferences ReferenceType*.

The *TargetNode* of a *Reference* of the *HasIOSignalRef ReferenceType* is providing the IO signal description for the *SourceNode*.

The *SourceNode* of *References* of this type shall be an *FdtParameter*.

The *TargetNode* of this *ReferenceType* shall be an *FdtIoSignalInfoType*.

The *HasIOSignalRef* is formally defined in Table 56.

Table 56 – HasIOSignalRef definition

Attributes	**Value**		
BrowseName	HasIOSignalRef		
InverseName	IsParameterOfIOSignal		
Symmetric	False		
IsAbstract	False		
References	**NodeClass**	**BrowseName**	**Comment**
Subtype NonHierarchicalReferences defined in IEC 62541-5.			

11 Mapping of DataTypes

11.1 Primitive data types – DeviceHealthEnumeration

The datatype *DeviceHealthEnumeration*[2] shall be mapped to the FDT2 datatype *DeviceStatus*. The values shall be mapped as defined in Table 57.

Table 57 – Mapping for DeviceHealthEnumeration

DeviceHealthEnumeration Value[a)]	**FDT DeviceStatus.StatusFlag Value**
NORMAL_0	Ok
FAILURE_1	Invalid
CHECK_FUNCTION_2	FunctionCheck
OFF_SPEC_3	OutOfSpecification
MAINTENANCE_REQUIRED_4	MaintenanceRequired
[a)] DeviceHealthEnumeration is explained in [1].	

11.2 Mapping to OPC DI types

11.2.1 Device type

11.2.1.1 General

IEC 62541-100 and IEC 62453 use different approaches in regard to representation of device type specific data. Some of the data defined in OPC UA for representation of the device type is available only in instance-specific data of the DTMs (see Figure 9).

The data from FDT may be mapped either to *Attributes* (table header "Attribute") or to properties and other child *Nodes* (table header "BrowseName") of the corresponding OPC UA *Node*.

2 DeviceHealthEnumeration is explained in [1].

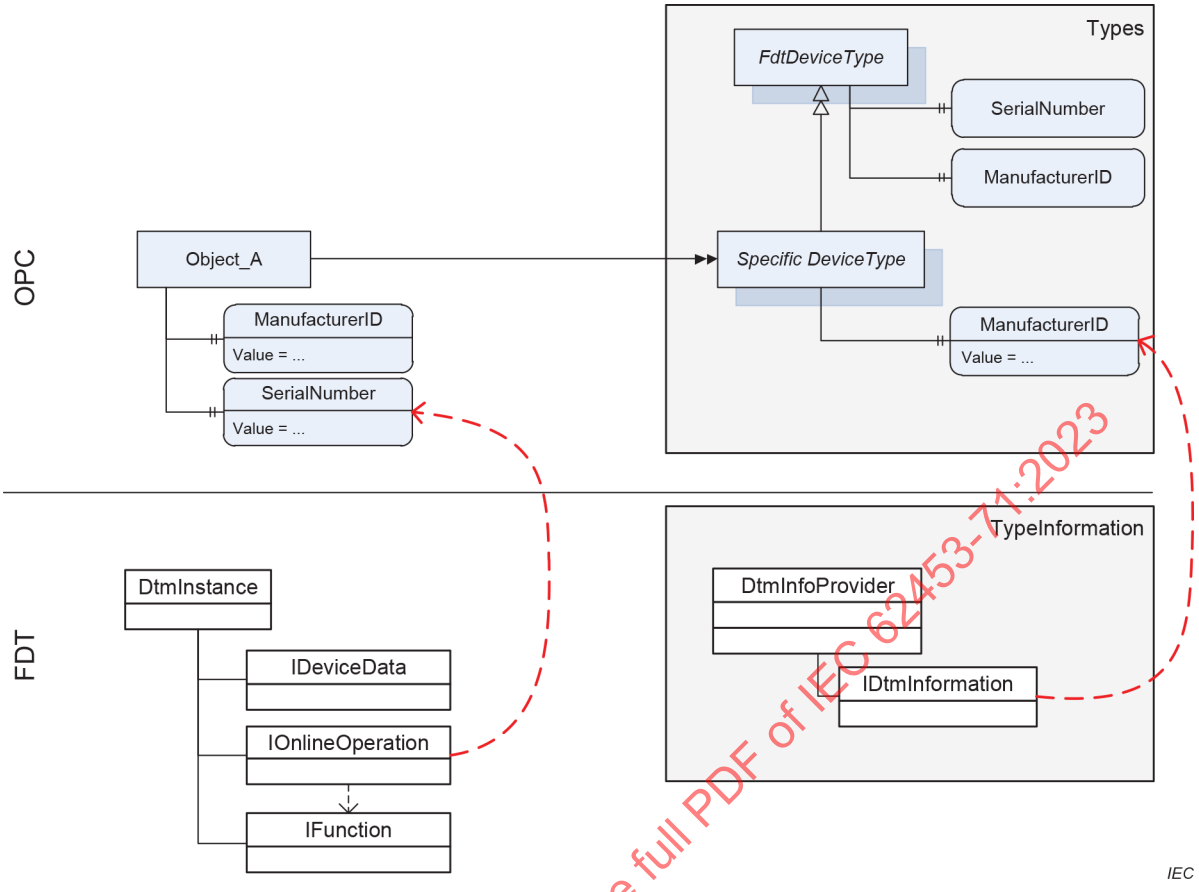


Figure 9 – Example for sources of DeviceType information

For this reason the device type specific data of OPC is mapped to data on several interfaces of the DTM (see Table 58).

11.2.1.2 Mapping for DeviceType ("Types" standard Object)

All device types provided by DTMs in IDtmInformation shall be represented as subtypes of *FdtDeviceType* in the "Types" standard Object (see Table 58).

Table 58 – DeviceType mapping

OPC	FDT			
Attribute	**Interface**	**Method**	**Data member**	**Description**
BrowseName	IDtmInformation	<>GetSupported-Types	DeviceTypeInfo.Name	
DisplayName	IDtmInformation	<>GetSupported-Types	DeviceTypeInfo.Name	
BrowseName	**Interface**	**Method**	**Data member**	**Description**
2:SerialNumber				Mapping for online device only, see Table 61
2:RevisionCounter				Not supported. Value is always set to -1
2:Manufacturer	IDtmInformation	<>GetSupported-Types	DeviceTypeInfo.ProductManufacturerName	
ManufacturerId	IDtmInformation	GetDeviceIdentInfo	DeviceIdentInfo.ManufacturerId	
2:Model	IDtmInformation	<>GetSupported-Types	DeviceTypeInfo.ProductName	
2:DeviceManual				All documents will be provided in folder Documentation
2:DeviceRevision	IDtmInformation	<>GetSupported-Types	DeviceTypeInfo.ProductRevision	
2:SoftwareRevision	IDtmInformation	GetDeviceIdentInfo	DeviceIdentInfo.SoftwareRevision	
2:HardwareRevision	IDtmInformation	GetDeviceIdentInfo	DeviceIdentInfo.HardwareRevision	
2:DeviceClass(O)	IDtmInformation	<>GetSupported-Types	DeviceTypeInfo.DeviceClassifications[0].DomainId + ":" + DeviceTypeInfo.DeviceClassifications[0].Id	The value is a concatenation of the enumeration member names for DomainId and Id.
DeviceTypeId	IDtmInformation	<>GetSupported-Types	DeviceTypeInfo.Id	
2:ManufacturerUri(O)				Not supported.
2:ProductCode(O)				Not supported.
2:ProductInstanceUri(O)				Not supported.
<CP_Identifier>(O)	IDtmInformation	<>GetSupported-Types	TypeInfo.BusCategories[].Category¬Type == required → <Identifier> = TypeInfo.BusCategories[].Protocol¬Name	HasComponent-reference of type RequiredProtocol.
	IDtmInformation IChannels	<>GetSupported-Types IChannels.CommunicationChannels	TypeInfo.BusCategories[].Category¬Type == supported → <Identifier> = CommunicationChannelItem.Id	HasComponent-reference of type SupportedProtocol. If one Channel requires multiple protocols, multiple connection points are provided, an additional number is concatenated to the channel id.
0:Icon	IDtmInformation	GetFdtIcon()	Icon	see explanation below the table
Applied from IFdtDeviceHealthType				
2:DeviceHealth				Mapping for online device only, see Table 61
2:DeviceHealthAlarms(O)	-	-	-	No mapping defined.

OPC	FDT			
BrowseName	**Interface**	**Method**	**Data member**	**Description**
Applied from IFdtSupportInfoType				
2:DeviceTypeImage (O)				This folder will contain items as defined in 11.2.8.1.
2:Documentation (O)	IDtmInformation	<>GetSupported-Types	DeviceTypeInfo.Documents	This folder will contain items that represent the shown FDT data member as defined in 6.3
2:ProtocolSupport (O)	INetworkData	GetNetworkDataInfo	NetworkData. DeviceInformationDocuments	This folder will be provided as defined in 11.2.8.2.
2:ImageSet(O)				Not supported.

A DTM can provide several icons (list of icons). The rule for selection of the icon and extraction of an image (a specific resolution) from the icon is frame application specific. The selection of the image format in general is implementation specific, all types are allowed that are defined in IEC 62541-3.

11.2.1.3 Mapping for Offline Device ("Objects" standard Object)

The information for a device (instance) is based on the information provided by a DTM (instance). The mapping for the device node is defined in Table 59.

Table 59 – Device information mapping

OPC	FDT			
Attribute	**Interface**	**Method**	**Data member**	**Description**
TypeReference	-	-	-	NodeId of the DeviceType
BrowseName	IDtm	DtmSystemGuiLabel		
DisplayName	IDtm	DtmSystemGuiLabel		

The device parameters provide offline identification information based on DTM type information intended for identification. This data may provide values or regular expressions to define ranges of supported values.

In the call to IDtmInformation.GetDeviceIdentInfo() a protocol must be specified. The first entry from the list of INetworkData.ActiveProtocols shall be used.

GetDeviceIdentInfo() may return a list of DeviceIdentInfo where the first entry is used for the parameters as defined below. Exposing the remaining list entries is implementation specific.

Table 60 – Offline device parameter mapping

OPC	FDT			
BrowseName	**Interface**	**Method**	**Data member**	**Description**
ManufacturerId	IDtmInformation	GetDeviceIdentInfo()	DeviceIdentInfo[0].GetManufacturerId()	
DeviceTypeId		IDtmInformation	GetDeviceIdentInfo()	DeviceIdentInfo[0].GetDeviceTypeId()
2:NetworkAddress	INetworkData	GetAddressInfo()	AddressInfo.DeviceAddresses[].Address	string array containing all addresses provided
DeviceTag	-/-	-/-	(default value)	
2:Serial Number	-/-	-/-	(default value)	
2:SoftwareRevision	IDtmInformation	GetDeviceIdentInfo()	DeviceIdentInfo[0].GetSoftwareRevision()	
2:HardwareRevision	IDtmInformation	GetDeviceIdentInfo()	DeviceIdentInfo[0].GetHardwareRevision()	
Contents of FunctionalGroup "Identification" (see 11.2.5.)				
ProtocolId	IDtmInformation	GetDeviceIdentInfo()	DeviceIdentInfo[0].GetProtocolId()	
ProtocolSpecific-Identification	IDtmInformation	GetDeviceIdentInfo()	DeviceIdentInfo[0].GetProtocolSpecificProperties()	
DeviceSpecific-Identification	IDtmInformation	GetDeviceIdentInfo()	DeviceIdentInfo[0].DeviceSpecificProperties	

11.2.1.4 Mapping for Online Device ("Objects" standard Object→ online reference)

Device parameters provide identification information based on data read online from the device. This data may be different from the data provided in the *DeviceType*.

Table 61 – Online device parameter mapping

OPC	FDT			
BrowseName	**Interface**	**Method**	**Data member**	**Description**
ManufacturerId	IHardwareInformation	<>HardwareScan()	DeviceScanInfo.GetManufacturerId()	
DeviceTypeId	IHardwareInformation	<>HardwareScan()	DeviceScanInfo.GetDeviceTypeId()	
2:NetworkAddress	IHardwareInformation	<>HardwareScan()	DeviceScanInfo.GetAddress()	
DeviceTag	IHardwareInformation	<>HardwareScan()	DeviceScanInfo.GetTag()	
2:DeviceHealth	IOnlineOperation	<>ReadDevice-Status()	DeviceStatus.StatusFlag	For mapping of the enumeration values see 11.1.
2:Serial Number	IHardwareInformation	<>HardwareScan()	DeviceScanInfo.SerialNumber()	
2:SoftwareRevision	IHardwareInformation	<>HardwareScan()	DeviceScanInfo.GetSoftwareRevision()	
2:HardwareRevision	IHardwareInformation	<>HardwareScan()	DeviceScanInfo.GetHardwareRevision()	
Contents of FunctionalGroup "Identification" (see 11.2.5.)				
ProtocolId	IHardwareInformation	<>HardwareScan()	DeviceScanInfo.GetProtocolId()	
ProtocolSpecific-Identification	IHardwareInformation	<>HardwareScan()	DeviceScanInfo.GetProtocolSpecificProperties()	
DeviceSpecific-Identification	IHardwareInformation	<>HardwareScan()	DeviceScanInfo.DeviceSpecificProperties	

11.2.2 TopologyElementType

The information provided by a DTM with IFunction.FunctionInfo and IData.DataInfo is mapped into the *MethodSet* and to the *ParameterSet*. For both types of information (Commands and Parameters) also references in *FunctionalGroup* nodes are created, so that an OPC Client may represent a joint user interface (see Figure 10).

The *BrowseName* and *DisplayName* of the *FunctionalGroup*-Nodes are based on the grouping in IFunction.FunctionInfo (FunctionGroup) and IData.DataInfo (DataGroup).

Information about IO signals are provided in a folder ProcessDataSet, which contains *FdtIoSignalInfoType* nodes. *FdtIoSignalInfoType* nodes are referenced by the corresponding *FdtParameter* nodes.

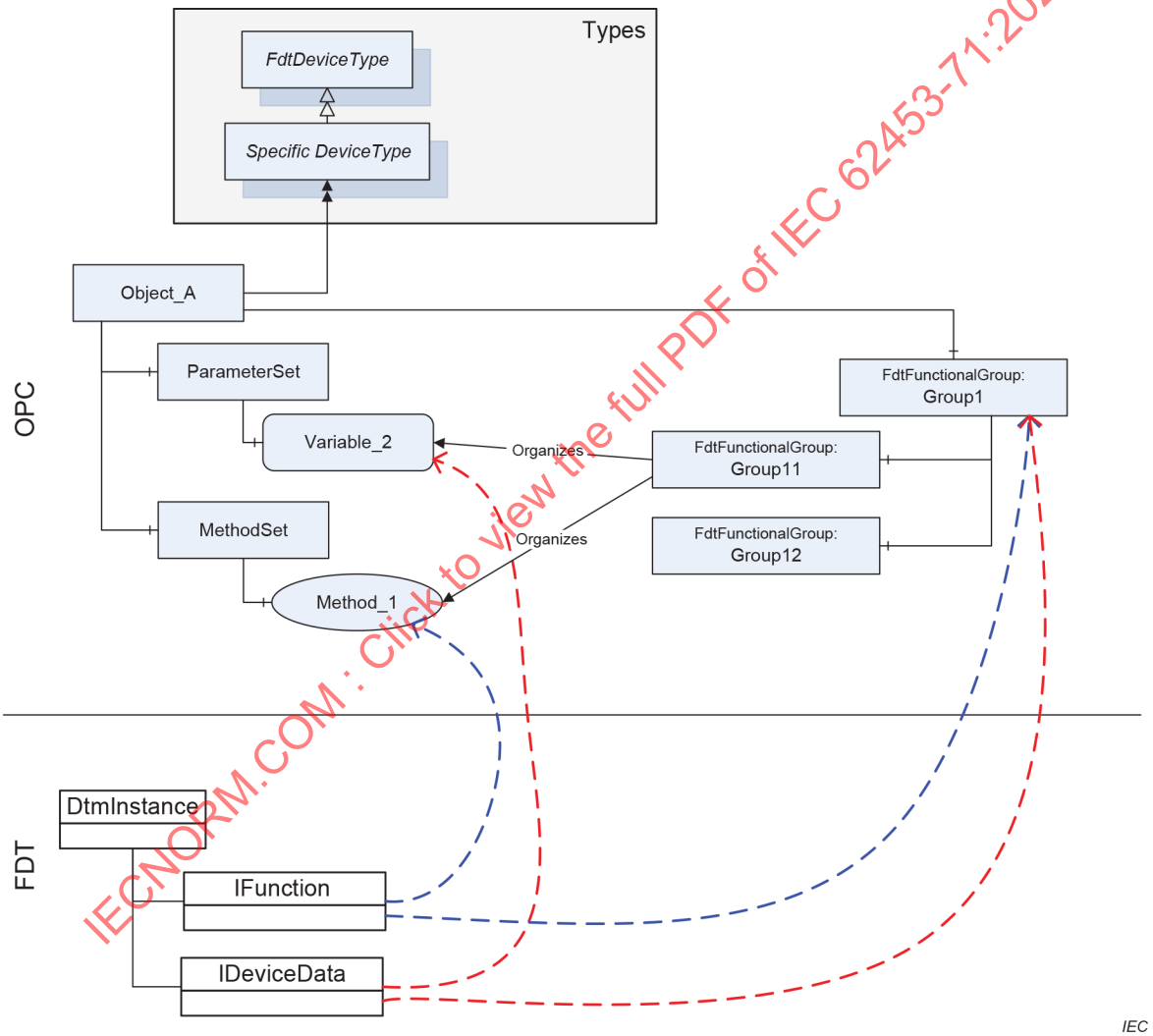


Figure 10 – Example for sources of TopologyType information

Table 62 – TopologyElementType mapping

OPC	FDT			
BrowseName	**Interface**	**Method**	**Data member**	**Description**
ParameterSet	IInstanceData / IDeviceData	<>GetDataInfo	DataInfo.DeviceDataItems	DeviceDataItems is a hierarchical list of data items that represent different types of data items. In the ParameterSet only AccessibleData and StructDataGroup shall be represented (see 11.2.7). The flat list of parameters is constructed from the hierarchical list where duplicates must be removed.
MethodSet	IFunction	FunctionInfo	FunctionInfo.FunctionItems	FunctionItems provides a hierarchical list of functions that are supported by the DTM. Only functions without user interface (CommandFunction) shall be represented in the MethodSet (see 11.2.6).
ProcessDataSet	IProcessData	<>GetProcessData	ProcessDataInfo.ProcessDataItems	ProcessDataItems provides a list of process data supported by the device. The FdtSignalInfo objects are referenced by the respective parameter objects (in the online device).
<GroupIdentifier>	IInstanceData / IDeviceData	<>GetDataInfo	DataInfo.DeviceDataItems	The hierarchical list provided through DeviceDataItems is mapped to a tree of FunctionalGroups. The strings used for the placeholder <GroupIdentifier> are defined by the names of the DataGroups.
Identification				The contents of the FunctionalGroup is defined in 11.2.5.
Lock				(defined in UA Devices)

11.2.3 FunctionalGroupType

The *FunctionalGroupType* is used to organize Parameters and Methods (which are defined in *ParameterSet* and *MethodSet*) in a user-friendly way (see Table 63). *FunctionalGroups* may be organized hierarchically by organizing *FunctionalGroups* as child nodes of *FunctionalGroups*.

Table 63 – FunctionalGroupType mapping

OPC	FDT			
Attribute	**Interface**	**Method**	**Data member**	**Description**
<GroupIdentifier>	IInstanceData / IDeviceData	<>GetDataInfo	DataInfo.DeviceDataItems == DataGroup → DataGroup.Name	Only DataInfo.DeviceDataItems shall be mapped, that are of type DataGroup. The Name member of the DataGroup shall be used as <GroupIdentifier>.
DisplayName	IInstanceData / IDeviceData	<>GetDataInfo	DataGroup.Label	
Description	IInstanceData / IDeviceData	<>GetDataInfo	DataGroup.Descriptor	
BrowseName	**Interface**	**Method**	**Data member**	**Description**
<ParameterIdentifier>	IInstanceData / IDeviceData	<>GetDataInfo	DataInfo.DeviceDataItems == Data → Data.Name	Only DataInfo.DeviceDataItems shall be mapped, that are of type Data. The Name member of the Data shall be used as <ParameterIdentifier>.
<MethodIdentifier>				(no mapping defined)
UIElement				(no mapping defined)

11.2.4 Identification FunctionalGroup

The Identification *FunctionalGroup* organizes references to parameters, which may be used to identify the device and its device type. The list of parameters that shall be provided in the Identification FunctionalGroup is defined in Table 64.

Table 64 – Mapping for FunctionalGroup Identification

References	BrowseName	Modelling Rule
Organizes	ManufacturerId	O
Organizes	DeviceTypeId	O
Organizes	2:NetworkAddress	O
Organizes	DeviceTag	M
Organizes	2:Serial Number	M
Organizes	2:SoftwareRevision	M
Organizes	2:HardwareRevision	M
Organizes	2:ProtocolId	OP
HasComponent	ProtocolSpecific¬Identification	O
HasComponent	DeviceSpecific¬Identification	O

The values of the referenced *Variables* may vary for online and offline device. See 11.2.1 for the mapping of these *Variables*.

Protocol specific properties shall be organised in a *FunctionalGroup* called "ProtocolSpecificIdentification" contained in the Identification *FunctionalGroup*.

Device specific properties shall be organised in a *FunctionalGroup* called "DeviceSpecificIdentification" contained in the Identification *FunctionalGroup*.

11.2.5 Device data and device methods

While device data and methods related to the device are defined in separate areas of the OPC UA Information model (i.e. *ParameterSet* and *MethodSet*), they are organized by functional groups into a concise representation, which may be structured to present different aspects of the device. A DTM provides different interfaces (e.g. IDeviceData and IFunction), where data and methods are presented independently, while each interface may provide an own structure.

In the mapping from DTM interfaces to OPC UA information model, the representation hierarchies of IData interfaces and IFunction interface are combined into *FunctionalGroups* to provide a concise representation of the device data and methods within one hierarchy.

This means, that the data of the IData interfaces are mapped to the *ParameterSet* node in the OPC UA Information model and the command functions from IFunction interface are mapped to the *MethodSet* of the device. The hierarchical structures in the IData and IFunction interfaces are mapped to the hierarchical structure provided by the *FdtFunctionalGroup* elements, which organize the representation of data and methods (see Figure 11).

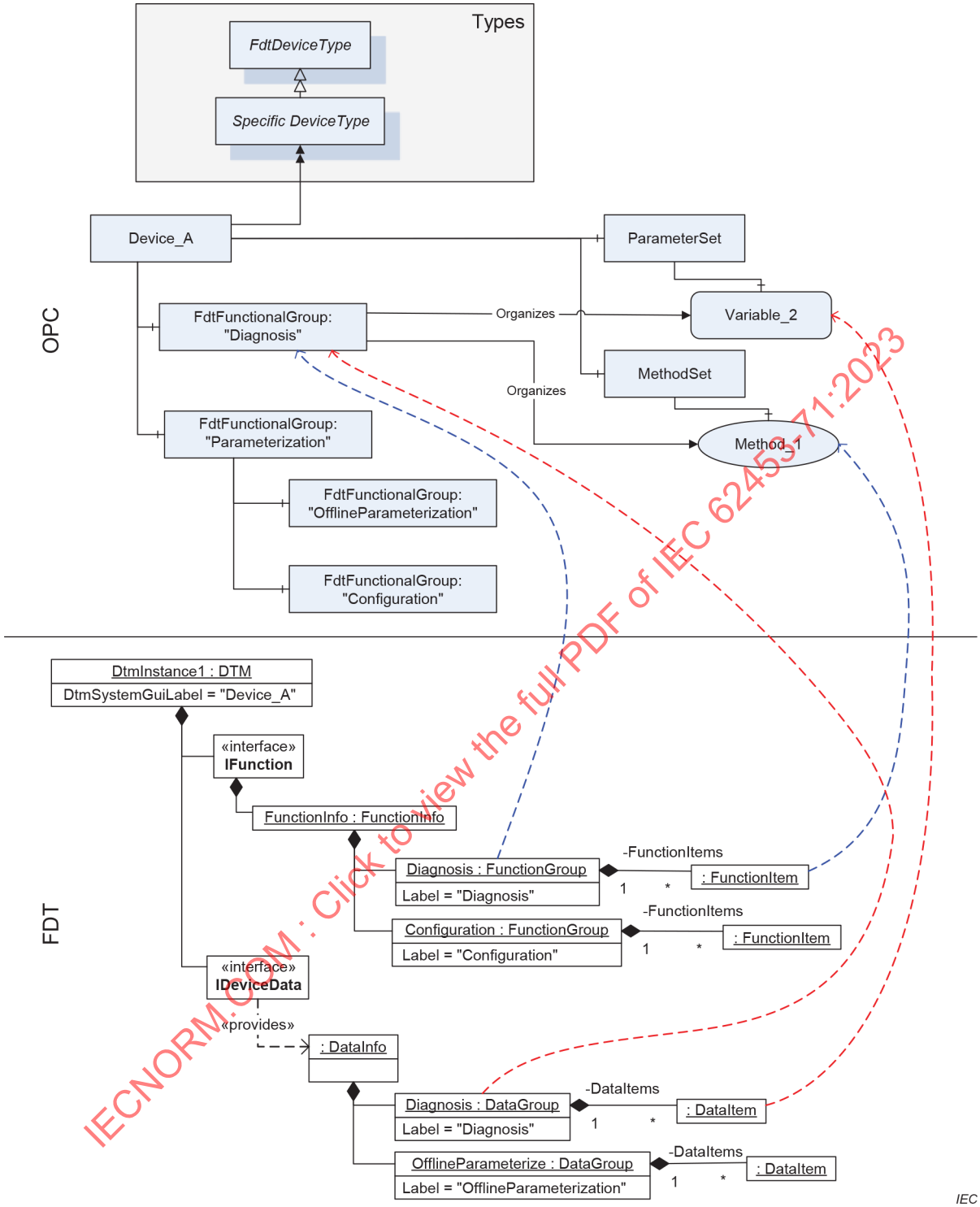


Figure 11 – Example for mapping of data and function information

11.2.6 Methods

11.2.6.1 General

A DTM instance can offer command functions. These functions do not have a user interface, they can have arguments and optional default values.

Each command function shall be mapped to a method in the *MethodSet* of the device instance (see Figure 12). The arguments are mapped to the InputArguments and OutputArguments for the methods. Optional arguments shall be included.

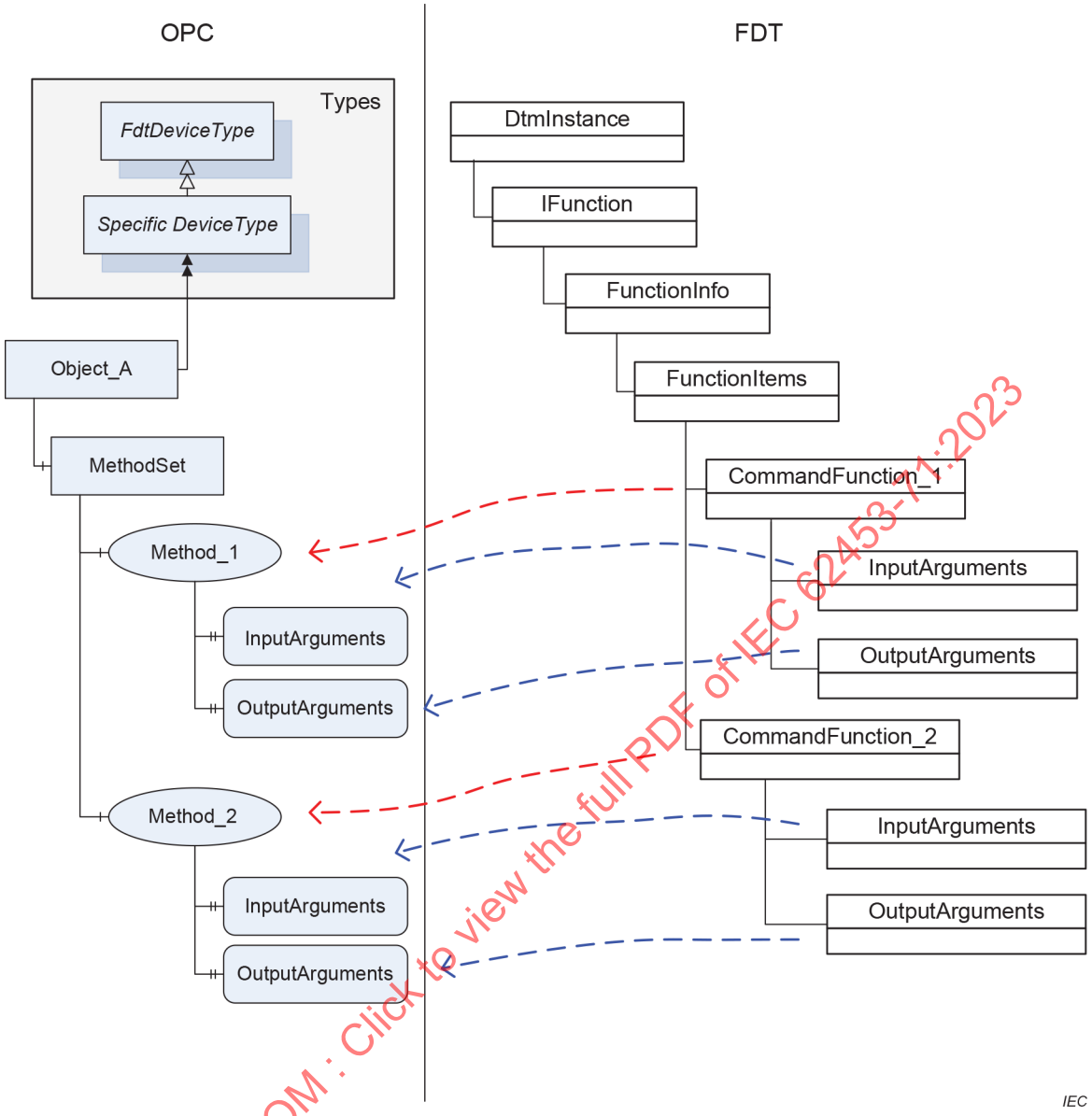


Figure 12 – Example for source of function information

Table 65 – Method node information mapping

OPC	FDT			
Attribute	**Interface**	**Method**	**Data member**	**Description**
BrowseName	IFunctions	FunctionInfo	FunctionItem.Label	
DisplayName	IFunctions	FunctionInfo	FunctionItem.Label	
Description	IFunctions	FunctionInfo	FunctionItem.Descriptor	

The DTMInfoBuilder can offer information about static functions that are independent of a DTM instance. Such FDT static functions may be mapped to methods on the device type created for the DTM. The execution of the method on a device object will need to be mapped to the execution of a static function with a StaticFunctionProvider. The Frame/Server will need to hold the reference to the provider internally and use it for execution of the method (see Figure 13).

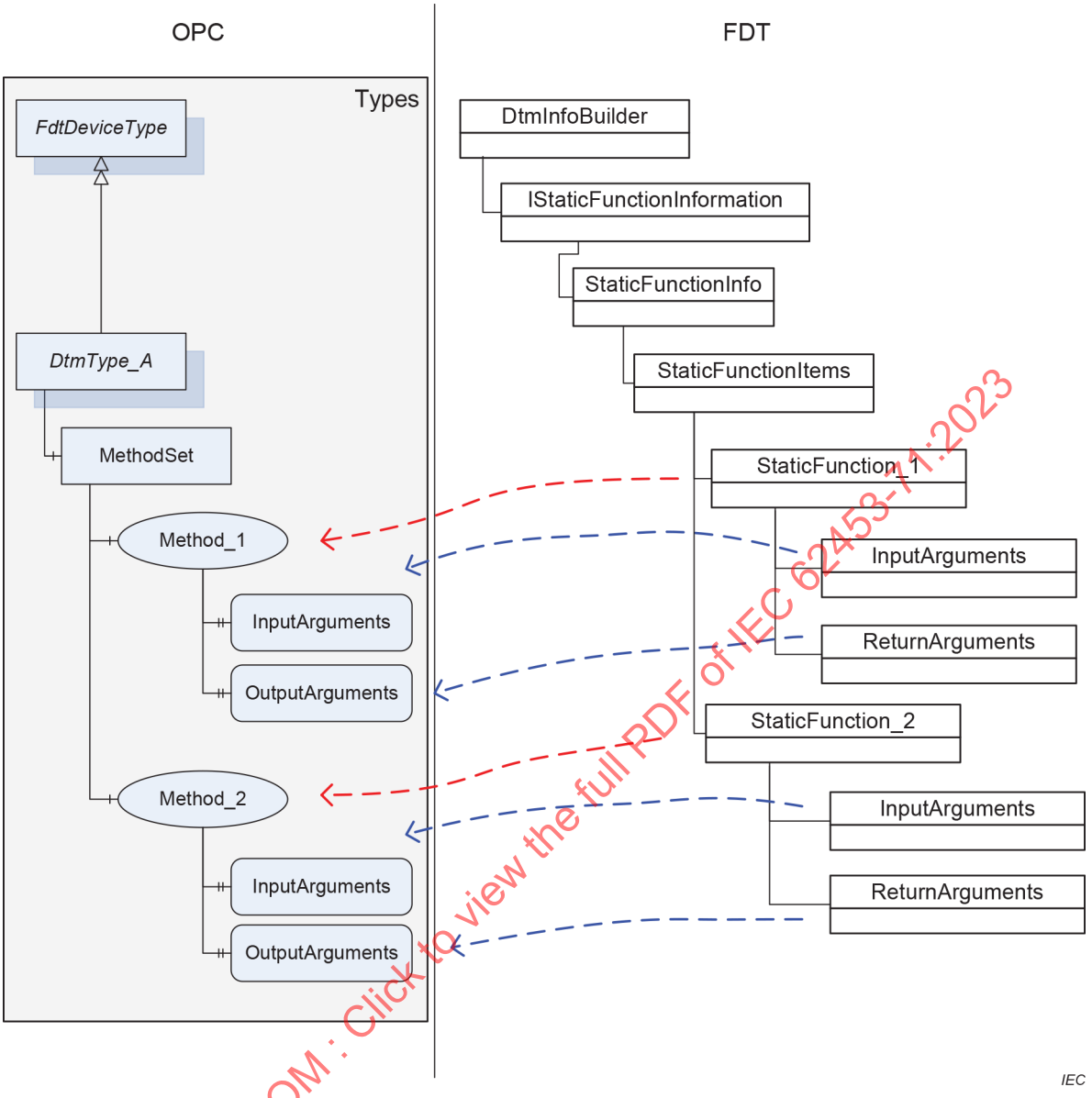


Figure 13 – Example for source of static function information

Table 66 – Method node information mapping for static function

OPC	FDT			
Attribute	**Interface**	**Method**	**Data member**	**Description**
BrowseName	IStaticFunctionInformation	GetStaticFunctions	StaticFunctionDescription.Label	
DisplayName	IStaticFunctionInformation	GetStaticFunctions	StaticFunctionDescription.Label	
Description	IStaticFunctionInformation	GetStaticFunctions	StaticFunctionDescription.Descriptor	

11.2.6.2 TransferServices

For a DTM instance it is mandatory to provide the interface IOnlineOperation if the device provides online data. This interface is mapped to an object of type *FdtTransferServiceType* (see 6.8). Since the DTM interface is not mandatory for all devices, the availability of the TransferServices depends on the availability of the DTM interface. The member SupportedTransfer indicates, which TransferServices are supported for the device.

Table 67 – TransferService mapping

OPC	FDT			
BrowseName	**Interface**	**Method**	**Data member**	**Description**
2:TransferToDevice	IOnlineOperation	BeginWriteDataToDevice / EndWriteDataToDevice	-	
2:TransferFromDevice	IOnlineOperation	BeginReadDataFromDevice / EndReadDataFromDevice	-	
2:FetchTransferResultData	IOnlineOperation	-	-	Retrieves the result / status from the executed methods.
SupportedTransfer	IOnlineOperation	-	SupportedTransfer	

The TransferServices Type object (with browse name "Transfer") is provided as child node of an offline device node (not for an online device node).

For the method FetchTransferResultData the server is allowed to limit the TransferResultData in regard to the list of transferred parameters (parameterDefs) to zero data. The results of the transfer will be represented in the parameter values of the device node.

11.2.7 Variable

11.2.7.1 General

The parameter set of a device (offline and online) is described by means of the DataInfo data type (available with the <GetDataInfo> method from IInstanceData (offline) and IDeviceData (online)).

DataInfo provides a description of the available data without the values. The actual values are accessible with <Read> and <Write> methods from IInstanceData (offline) and IDeviceData (online).

DataInfo provides a list of data items (member DeviceDataItems), which may represent a range of types. Some of these data items are mapped to DataVariables (see Table 68).

Table 68 – Mapping of FDT data items

OPC	FDT
DataVariable (according to IEC 62541-3)	Data
	UnitData
	RangeData
	AlarmData
	SubstituteData
	StructDataGroup
FunctionalGroup (according to IEC 62541-100)	DataGroup
Block object (according to IEC 62541-100)	ModuleDataGroup

11.2.7.2 FdtParameter

11.2.7.2.1 General

Table 69 shows the mapping for the FdtParameter.

Table 69 – FdtParameter mapping

OPC	FDT			
Attribute	**Interface**	**Method**	**Data member**	**Description**
NodeClass				Fixed value: Variable
BrowseName	IData	<GetDataInfo>	Name	
DisplayName	IData	<GetDataInfo>	Label	
Description	IData	<GetDataInfo>	Descriptor	
WriteMask				Generated by Frame
UserWriteMask	IData	<GetDataInfo>	IsChangeEnabled	Generated by Frame
Value	IData	<Read> / <Write>		
DataType	IData	<GetDataInfo>	DataTypeInfo	
ValueRank	IData	<GetDataInfo>	DataTypeInfo	Indicates whether the datatype is an array.
ArrayDimensions	IData	<GetDataInfo>	DataTypeInfo	If the datatype is an array, then provide dimensions.
AccessLevel	IData	<GetDataInfo>	IsReadable, IsWritable	Generated by Frame
UserAccessLevel	IData	<GetDataInfo>	IsReadable, IsWritable	Generated by Frame
MinimumSamplingInterval				Defined by Frame.
Historizing				Defined by Frame.
BrowseName				
2:EURange	IData	<GetDataInfo>	RangeDataRefs	
2:EngineeringUnits	IData	<GetDataInfo>	UnitDataRef	
0:HasComponent				May be used for structured datatypes (StructDataGroup)
2:HasDefinition				Defined by Frame.
2:ValuePrecision				Defined by Frame.
DisplayFormat	IData	<GetDataInfo>	DisplayFormat	
AlarmType	IData	<GetDataInfo>	AlarmType	
RangeType	IData	<GetDataInfo>	RangeType	
SubstitutionType	IData	<GetDataInfo>	SubstitutionType	
ApplicationId	IData	<GetDataInfo>	ApplicationId	
SemanticInfo	IData	<GetDataInfo>	SemanticInfos	
DataRef	IData	<GetDataInfo>	DataRefs	
IOSignalRef	IData	<GetDataInfo>	IOSignalRef	
AlarmDataRef	IData	<GetDataInfo>	AlarmDataRefs	
SubstituteDataRef	IData	<GetDataInfo>	SubstituteDataRef	

11.2.7.2.2 Datatype mapping

The datatype mapping for parameter data is described in Table 70.

Table 70 – Mapping of simple data types

OPC	FDT	
Data type	**Data type**	**Description**
Float	Float	
Double	Double	
Byte	Byte	
Int32	Int	
Int64	Long	
UInt32	UInt	
UInt64	ULong	
DateTime	DateTime	
DateTime	Date	Date is represented as date at 0:00 o'clock.
Duration	Time	Time is expressed as duration since midnight.
Duration	TimeSpan	Duration is expressed in milliseconds, TimeSpan is expressed in terms of days, hours, minutes. For the conversion of large TimeSpan a loss of precision may occur.
String	String	
Array of Byte	BinaryByteArray	
Array of Bit	BinaryBitArray	
Enumeration	Enumerator	
Boolean	Boolean	
SByte	SByte	

In OPC UA array datatypes are represented by a combination of the properties datatype, value-rank and array-dimensions (see IEC 62541-5).

In FDT array datatypes are represented with a dedicated ArrayDatatypeInfo datatype. This datatype has a member ArrayDatatypeInfo.ArrayDimensions, which specifies the length of each dimension of the array.

The mapping for array data types is defined as follows.

– OPC datatype value is mapped according to Table 70.
– OPC value rank is:
  - -1 for a simple datatype, or
  - Number of fields in FDT ArrayDatatypeInfo.ArrayDimensions (always > 0)
– OPC array-dimensions property is:
  - NULL for a simple datatype, or
  - has the same value as ArrayDatatypeInfo.ArrayDimensions.
    – value of 0 means that the array has a variable length,
    – values > 0 indicate a defined length.

11.2.8 Device support information

11.2.8.1 Device Type Image

All Bitmaps provided by a DTM shall be represented in the *DeviceTypeImage* folder according IEC 62541-100 (see Table 58). An image shall be represented by an *FdtParameter* with *DataType* set to one of the defined image data types. The actual image is provided by the value attribute as ByteString.

The mapping for a device type image is defined in Table 71.

Table 71 – Device Type Image mapping

OPC	FDT			
BrowseName	**Interface**	**Method**	**Data member**	**Description**
SemanticInfo	IDtmInformation	<>GetSupportedTypes	TypeInfo.Bitmaps.SemanticInfo	
Bitmap	IDtmInformation	GetBitmap()	Bitmap	

Selection of the image format according to FDT2 is specific for a Frame Application. All image types are allowed that are defined in IEC 62541-3.

11.2.8.2 Protocol support files

Protocol support files provided for a device are exposed as *Variables* organised in the *ProtocolSupport* folder. They may represent various types of information as defined by a protocol. Examples are a GSD or a CFF file. The *ProtocolSupport* folder is formally defined in IEC 62541-100:2015, 5.7.3. The mapping is protocol independent. All the documents listed in NetworkData.DeviceInformationDocuments are mapped. The value of the <PlaceHolder> is defined by the label of the respective document (Document.Label).

Table 72 – ProtocolSupport mapping

OPC	FDT			
BrowseName	**Interface**	**Method**	**Data member**	**Description**
<PlaceHolder>	INetworkData	GetNetworkDataInfo	NetworkData. DeviceInformationDocuments	This applies to all protocols. The optional data member provides protocol support files for all protocols.

The label of the DeviceInformationDocument shall be used as *BrowseName* for the OPC Node.

11.2.8.3 FdtIoSignalInfoType

All IOSignalInfo provided by the DTM in the IProcessData interface are mapped to respective FdtIoSignalInfo objects.

Table 73 – FdtIoSignalInfoType node information mapping

OPC	FDT			
Attribute	**Interface**	**Method**	**Data member**	**Description**
BrowseName	IProcessData	<>GetProcessData()	IOSignalInfo.Name	
DisplayName	IProcessData	<>GetProcessData()	IOSignalInfo.Label	
Description	IProcessData	<>GetProcessData()	IOSignalInfo.Descriptor	
BrowseName				
FrameApplicationTag	IProcessData	<>GetProcessData()	IOSignalInfo.FrameApplicationTag	
IECDatatype	IProcessData	<>GetProcessData()	IOSignalInfo.IECDatatype	
IsLocked	IProcessData	<>GetProcessData()	IOSignalInfo.IsLocked	
IsSafety	IProcessData	<>GetProcessData()	IOSignalInfo.IsSafety	
RoutedIoSignalId	IProcessData	<>GetProcessData()	IOSignalInfo.RoutedIoSignalId	
SemanticInfo	IProcessData	<>GetProcessData()	IOSignalInfo.SemanticInfo	
SignalType	IProcessData	<>GetProcessData()	IOSignalInfo.SignalType	
HasAlarmInfo	IProcessData	<>GetProcessData()	IOSignalInfo.AlarmInfoRef	
HasDeviceData	IProcessData	<>GetProcessData()	IOSignalInfo.DeviceDataRef	
HasRange	IProcessData	<>GetProcessData()	IOSignalInfo.RangeInfoRef	
HasSubstituteValue	IProcessData	<>GetProcessData()	IOSignalInfo.SubstituteValueRef	
HasUnit	IProcessData	<>GetProcessData()	IOSignalInfo.UnitInfoRef	

11.2.9 FdtProtocolType

The *FdtProtocolType* and its sub-types are used to specify a communication protocol that is supported by a Device or Network. The *BrowseName* of each instance of a *ProtocolType* shall define the Communication Profile (see IEC 62541-100:2015, 5.9).

OPC UA for FDT uses a sub-type *FdtProtocolType* (see 6.7) for mapping the needed information.

Table 74 – FdtProtocolType node information mapping

OPC	FDT			
Attribute	**Interface**	**Method**	**Data member**	**Description**
BrowseName	IDtmInformation	<>GetSupported-Types	DeviceTypeInfo.BusCategories[].protocolName	
DisplayName	IDtmInformation	<>GetSupported-Types	DeviceTypeInfo.BusCategories[].protocolName	
BrowseName				
BusCategory	IDtmInformation	<>GetSupported-Types	DeviceTypeInfo.BusCategories[].protocolId	The protocolId is formatted as string with upper case letters.

12 Profiles and Conformance Units

12.1 Conformance Units

Table 75 defines the applicable *ConformanceUnits* for the OPC UA Information Model for FDT. These *ConformanceUnits* are specified according to IEC 62541-7.

Table 75 – Conformance Units for FDT

Category	Title	Description
Server		
Server	FDT DeviceType	Support of objects of specific types derived from FdtDeviceType.
Server	FDT FunctionalGroup Type	Support of the FdtFunctionalGroupType.
Server	FDT DeviceHealth interface	Support of the IFdtDeviceHealth interface as defined in this document.
Server	FDT SupportInfo interface	Support of the IFdtSupportInfo interface as defined in this document.
Server	FDT DocumentTypes	Support of the FdtDocumentType and derived types.
Server	FDT ProtocolType	Support of the FdtProtocolType.
Server	FDT TransferServiceType	Support of the FdtTransferServiceType
Server	FDT IOSignalInfo	Support of the FdtIoSignalInfoType.
Server	FDT Parameter	Support of the FdtParameter variable type.
Client		
Client	FDT Client DeviceType	Consumes objects of specific types derived from FdtDeviceType.
Client	FDT Client FunctionalGroup Type	Consumes objects of the FdtFunctionalGroupType.
Client	FDT Client DeviceHealth interface	Uses the IFdtDeviceHealth interface as defined in this document.
Client	FDT Client SupportInfo interface	Uses the IFdtSupportInfo interface as defined in this document.
Client	FDT Client DocumentTypes	Consumes objects of the FdtDocumentType and derived types.
Client	FDT Client ProtocolType	Consumes objects of the FdtProtocolType.
Client	FDT Client TransferServiceType	Consumes objects of the FdtTransferServiceType
Client	FDT Client IOSignalInfo	Consumes objects of the FdtIoSignalInfoType.
Client	FDT Client Parameter	Consumes objects of the FdtParameter variable type.

12.2 Profiles

12.2.1 Profile list

Table 76 lists all Profiles defined in this document and defines their URIs.

Table 76 – Profile URIs for FDT

Profile	URI
FDT Base Server Profile	http://opcfoundation.org/UA-Profile/FDT/Server/Base
FDT General Server Facet	http://opcfoundation.org/UA-Profile/FDT/Server/General
FDT General Client Facet	http://opcfoundation.org/UA-Profile/FDT/Server/Client

12.2.2 Server Facets

12.2.2.1 Overview

The following subclauses specify the *Facets* available for *Servers* that implement the FDT companion specification. Each subclause defines and describes a *Facet* or *Profile*.

12.2.2.2 FDT Base Server Profile

Table 77 defines a *Profile* that describes the basic requirements for an OPC UA server supporting the FDT information model.

Table 77 – FDT Base Server Profile

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	0:Embedded 2017 UA Server Profile http://opcfoundation.org/UA-Profile/Server/EmbeddedUA2017	M
Profile	0:Data Access Server Facet http://opcfoundation.org/UA-Profile/Server/DataAccess	M
Profile	2:BaseDevice_Server_Facet	M
Profile	2:DeviceIdentification_Server_Facet	M
Profile	2:DeviceCommunication_Server_Facet	O
Profile	2:DeviceIntegrationHost_Server_Facet	O
Profile	FDT General Server Facet	M
Subscription Services	0:Subscription Durable	M

12.2.2.3 FDT General Server Facet

Table 78 defines a *Facet* that describes the general requirements for a server.

Table 78 – FDT General Server Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
FDT	FDT DeviceType	M
FDT	FDT DeviceHealth interface	M
FDT	FDT ProtocolType	M
FDT	FDT Parameter	M
FDT	FDT FunctionalGroup Type	O
FDT	FDT SupportInfo interface	O
FDT	FDT DocumentTypes	O
FDT	FDT TransferServiceType	O
FDT	FDT IOSignalInfo	O

12.2.3 Client Facets

12.2.3.1 Overview

Table 79 specifies the *Facets* available for *Clients* that implement the FDT companion specification.

12.2.3.2 FDT General Client Facet

Table 79 defines a *Facet* that describes the base characteristics for all OPC UA *Clients* that make use of this companion specification. Additional *Profiles* will define support for various information models that are part of this document.

Table 79 – FDT General Client Facet

Group	Conformance Unit / Profile Title	Mandatory / Optional
Profile	0:AddressSpace Lookup Client Facet http://opcfoundation.org/UA-Profile/Client/AddressSpaceLookup	O
Profile	0:DataAccess Client Facet http://opcfoundation.org/UA-Profile/Client/DataAccess	M
Profile	0:DataChange Subscriber Client Facet http://opcfoundation.org/UA-Profile/Client/DataChangeSubscriber	O
Session Services	0:Session Client Detect Shutdown	M
FDT	FDT Client DeviceType	M
FDT	FDT Client Parameter	M
FDT	FDT Client FunctionalGroup Type	O
FDT	FDT Client IFdtDeviceHealth interface	O
FDT	FDT Client IFdtSupportInfo interface	O
FDT	FDT Client DocumentTypes	O
FDT	FDT Client ProtocolType	O
FDT	FDT Client TransferServiceType	O
FDT	FDT Client IOSignalInfo	O

13 Namespaces

13.1 Namespace metadata

Table 80 defines the namespace metadata for this document. The *Object* is used to provide version information for the namespace and an indication about static *Nodes*. Static *Nodes* are identical for all *Attributes* in all *Servers*, including the *Value Attribute*. See IEC 62541-5 for more details.

The information is provided as *Object* of type *NamespaceMetadataType*. This *Object* is a component of the *Namespaces Object* that is part of the *Server Object*. The *NamespaceMetadataType ObjectType* and its *Properties* are defined in IEC 62541-5.

The version information is also provided as part of the ModelTableEntry in the UANodeSet XML file. The UANodeSet XML schema is defined in IEC 62541-6.

Table 80 – NamespaceMetadata Object for this document

Attribute	Value	
BrowseName	http://opcfoundation.org/UA/FDT/	
Property	**DataType**	**Value**
0:NamespaceUri	0:String	http://opcfoundation.org/UA/FDT/
0:NamespaceVersion	0:String	1.01.00
0:NamespacePublicationDate	0:DateTime	2021-08-06
0:IsNamespaceSubset	0:Boolean	False
0:StaticNodeIdTypes	0:IdType []	0
0:StaticNumericNodeIdRange	0:NumericRange []	
0:StaticStringNodeIdPattern	0:String	

NOTE The *IsNamespaceSubset Property* is set to False as the UANodeSet XML file contains the complete Namespace. *Servers* only exposing a subset of the Namespace need to change the value to True.

13.2 Handling of OPC UA namespaces

Namespaces are used by OPC UA to create unique identifiers across different naming authorities. The *Attributes NodeId* and *BrowseName* are identifiers. A *Node* in the UA *AddressSpace* is unambiguously identified using a *NodeId*. Unlike *NodeIds*, the *BrowseName* cannot be used to unambiguously identify a *Node*. Different *Nodes* may have the same *BrowseName*. They are used to build a browse path between two *Nodes* or to define a standard *Property*.

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 *EngineeringUnits Property*. All *NodeIds* of *Nodes* not defined in this document shall not use the standard namespaces.

Table 81 provides a list of mandatory and optional namespaces used in an FDT OPC UA *Server*.

Table 81 – Namespaces used in a FDT Server

NamespaceURI	Description	Use
http://opcfoundation.org/UA/	Namespace for *NodeIds* and *BrowseNames* defined in the OPC UA specification. This namespace shall have namespace index 0.	Mandatory
Local Server URI	Namespace for nodes defined in the local server. This namespace shall have namespace index 1.	Mandatory
http://opcfoundation.org/UA/DI/	Namespace for *NodeIds* and *BrowseNames* defined in IEC 62541-100. The namespace index is *Server* specific.	Mandatory
http://opcfoundation.org/UA/FDT/1.01/	Namespace for *NodeIds* and *BrowseNames* defined in this document. The namespace index is *Server* specific.	Mandatory
Vendor specific types	A *Server* may provide vendor-specific types like types derived from *ObjectTypes* defined in this document in a vendor-specific namespace.	Optional
Vendor specific instances	A *Server* provides vendor-specific instances of the standard types or vendor-specific instances of vendor-specific types in a vendor-specific namespace. It is recommended to separate vendor specific types and vendor specific instances into two or more namespaces.	Mandatory

Table 82 provides a list of namespaces and their indices used for *BrowseNames* in this document. The default namespace of this document is not listed since all *BrowseNames* without prefix use this default namespace.

Table 82 – Namespaces used in this document

NamespaceURI	Namespace Index	Example
http://opcfoundation.org/UA/	0	0:EngineeringUnits
http://opcfoundation.org/UA/DI/	2	2:DeviceRevision

Annex A
(normative)

FDT namespace and identifiers

This appendix defines the numeric identifiers for all of the numeric *NodeIds* defined in this document. The identifiers are specified in a CSV file with the following syntax:

```
<SymbolName>, <Identifier>, <NodeClass>
```

Where the *SymbolName* is either the *BrowseName* of a *Type Node* or the *BrowsePath* for an *Instance Node* that appears in the specification and the *Identifier* is the numeric value for the *NodeId*.

The *BrowsePath* for an *Instance Node* is constructed by appending the *BrowseName* of the instance *Node* to the *BrowseName* for the containing instance or type. An underscore character is used to separate each *BrowseName* in the path. Let's take for example, the *FdtParameter ObjectType Node* which has the *DisplayFormat Property*. The *SymbolName* for the *DisplayFormat InstanceDeclaration* within the *FdtParameter* declaration is: *FdtParameter_DisplayFormat*.

The *NamespaceUri* for all *NodeIds* defined here is http://opcfoundation.org/UA/FDT/1.01/

The CSV released with this version of the specification can be found here:

http:/opcfoundation.org/UA/schemas/FDT/1.01/Opc.Ua.FDT.NodeSet.csv

NOTE 1 The latest CSV that is compatible with this version of the specification can be found here:
https://opcfoundation.org/UA/schemas/FDT/Opc.Ua.FDT.NodeSet.csv

A computer processible version of the complete Information Model defined in this document is also provided. It follows the XML Information Model schema syntax defined in IEC 62541-6.

The Information Model Schema for this version of the document (including any revisions, amendments or errata) can be found here:

http://www.opcfoundation.org/UA/schemas/FDT/1.01/Opc.Ua.FDT.NodeSet.xml

NOTE 2 The latest Information Model schema that is compatible with this version of the document can be found here:
http://www.opcfoundation.org/UA/schemas/FDT/Opc.Ua.FDT.NodeSet.xml

Annex B
(informative)

Use cases

B.1 General

The following use cases describe how base features are supported by the FDT OPC UA system architecture.

B.2 Use case: List topology

Use case	List Topology
Description	The OPC UA client shows the topology in a hierarchical tree to the user. The tree is provided by an OPC UA server for FDT OPC Information Model. If the OPC UA server supports a network topology, it is provided according to the 'Device Integration Host Model' defined in OPC UA DI.
Stakeholder	
Preconditions	Topology is available in OPC UA Server
Post conditions	User can see the topology in a hierarchy
Actors	Engineer, Expert, Observer
Trigger	Whenever a user needs a list of devices for further access
Frequency	
Description	
Step	Action
1	User triggers connect from OPC Client to OPC Server
2	User starts browsing from node 2:DeviceTopology
3	OPC Client displays the topology to the user
Exceptions	
Requirements	User has access permissions to the server
Notes	Network/communication topology Physical topology
Open Issues	

B.3 Use case: Identify device

Use case	Identify Device
Description	A user gets information about a physical device selected in a topology. The OPC UA client provides this information to the user by reading identification information provided according to the FDT OPC UA information model.
Preconditions	Topology and device node is available in OPC UA Server
Post conditions	
Actors	Engineer, Expert, Observer, Plant Asset Manager
Trigger	Whenever a user needs information about a device
Frequency	
Description	
Step	Action
1	User browses namespace to a node representing a device
2	User starts reading identification information of the device
3	OPC Server obtains information about the device from associated DTM
4	OPC Client displays the identification values to the user
Exceptions	
Requirements	User has access permissions to the server
Notes	
Open Issues	

B.4 Get list of available device parameters

B.4.1 Use case: Browse device parameters

Use case	Browse device parameters.
Description	OPC Client has the UI windows for displaying the device details information. If the device has the blocks, the UI window displays block names which the device has. If a device has blocks, the UI window displays parameter names which are included in blocks. If a device does not have a block, the UI window displays parameter names which the device has.
Stakeholder	Mass Configuration
Preconditions	Use Case B.2 (List Topology)
Post conditions	On the OPC client, user can see the parameter names which the device or the block has.
Actors	Engineer, Expert, Observer, Plant Asset Manager
Trigger	Whenever a user needs a list of parameters for further access.
Frequency	
Description	
Step	Action
1	On OPC Client, user opens a UI window of a device which is selected on the topology which is listed by the Use Case B.2.
2	User browses a name list on the selected device in the OPC Client.
3	OPC Server retrieves a name list of the selected device.
4	OPC Client displays a name list to user. If selected device has blocks (if the device is FF-H1 or PROFIBUS PA), OPC Client displays the block name list.
5	User selects an item in the name list, and user browses the selected item (block name list or parameter list).
6	If the object type of selected item is "Block", OPC Server retrieves available parameters of the selected block.
Extension	
Step	Branching Action
4	OPC Server gets the attributes (Label Name / Help Description) of the selected item in the list and displays in the OPC Client.
Variations	
Step	Branching Action
4	If selected device has no blocks (if the device is HART), OPC Client displays the parameter name list.
6	If the object type of selected item is "Parameter", OPC Server retrieves available attributes of the selected parameter. For the enumeration case, an OPC Client displays the corresponding label with the value.
Exceptions	Step 2, The device is not ready to communicate.
	Step 3, The DTM which is browsed does not provide a block list (if the device is FF-H1) or a parameter list.
	Step 6, If the object type of selected item is "Parameter Element", OPC Server does not return a list. OPC Client displays a message informing user that there are no available items.
Requirements	User has access permissions to the server
Notes	[Hierarchy of items] [Up to the parameter] FF-H1 / PROFIBUS PA: {Device} --- {Block} --- {Parameter} HART: {Device} --- {Parameter} [Up to the element] FF-H1: {Device} --- {Block} --- {Parameter} --- {Element(*)} (*) Element = the member variable of the structure [Up to the enumeration] FF-H1 / PROFIBUS PA: {Device} --- {Block} --- {Parameter} --- {Enumeration} FF-H1: {Device} --- {Block} --- {Parameter} --- {Element} --- {Enumeration} HART: {Device} --- {Parameter} --- {Enumeration}
Open Issues	

B.4.2 Use case: Get attributes of a device parameter

Use case	Get attributes of a device parameter
Description	OPC Client has the UI windows for displaying the device details information. The UI window displays parameter names, its' engineering units and values (if the parameter is readable). When OPC Client displays the value of a parameter, the value is displayed according to the "display format", which is described in the DD. If the OPC Client also has the parameter-write-function and if the parameter is writable, an UI-element for writing the value to a parameter is displayed.
Stakeholder	Mass Configuration
Preconditions	Topology and device node are available in OPC UA Server. Device is connected to the OPC UA Server. OPC Client gets the parameter list of a device from OPC Server. To display the parameter name and its value, OPC Client tries to get the detail information of each parameter.
Post conditions	On the OPC client, user can see the parameter names, and can see the detail information for each parameter. Detail Information of parameter: the data type, the label name, the minimum value, the maximum value, the engineering units, writable, readable, the enumeration list, the display format, the edit format.
Actors	Engineer, Expert, Observer, Plant Asset Manager
Trigger	Whenever a user needs the attributes of parameters for further access.
Frequency	
Description	
Step	Action
1	OPC Client requests to get the attributes of a parameter of a device to OPC Server.
2	OPC Server gets the attributes of the parameter which is specified by OPC Client from the device DTM.
3	OPC Client displays the parameter name to use the label name, and the engineering units.
Exceptions	Step 1, The specified parameter does not exist in the device.
Requirements	
Notes	To use the attributes of a device parameter which is gotten by this use case, OPC Client can work as follows. [For read parameter] OPC Client can recognize a parameter is readable or not, and then OPC Client gets the value of the parameter via OPC Server from the device. OPC Client displays the corresponding label with the value if the data type of a parameter is the enumeration, otherwise (if the data type is not the enumeration) OPC Client displays the value according to the "display format". [For write parameter] OPC Client can recognize a parameter is writable or not, and then OPC Client can check the entered value whether the data type, and the minimum value and the maximum value of the entered value are correct or not when user tries to write a value to a parameter.
Open Issues	

B.5 Use case: Get Device Status

Use case	Get Device Status
Description	DTM can provide the NE107 status as the device status by the next procedure. Reading the data from the device parameters which express the device status or the results of the diagnosis of the device. Calculating the bits and mapping to the NE107 status definition.
Stakeholder	Mass Configuration
Preconditions	Client connected to OPC Server.
Post conditions	OPC Client can display NE107 status as the device status.
Actors	Engineer, Expert, Observer, Plant Asset Manager
Trigger	Whenever a user needs the device status information of the device.
Frequency	
Description	
Step	Action
2	User triggers to retrieve device status of a device in OPC Client.
3	OPC Client requests to get the diagnosis parameter/s of a device to OPC Server.
4	OPC Server retrieves the device status from the DTM
5	DTM reads the value of the diagnosis parameter/s of a device and calculates the status
8	OPC Client displays the NE107 status of the device.
Exceptions	Step 4, No diagnosis parameter/s are defined, if the DTM does not support to retrieve the diagnosis parameter/s.
Requirements	
Notes	
Open Issues	

B.6 Use case: Get Device Diagnostics

Use case	Get Device Diagnostic Information
Description	A software or user requests diagnostic information (context specific diagnostic information) from a physical device. The OPC UA client collects this information by reading device diagnostic information provided according to the OPC UA DI information model.
Stakeholder	
Used by	
Preconditions	Device node is available in OPC UA Server Device is connected to the OPC UA Server (means online or only somehow accessible?) Use case B.2 (List Topology)
Post conditions	Maintenance or asset manager can derive actions
Actors	Engineer, Expert, Observer, Maintenance Manager, Asset Manager, Operator
Trigger	Status of device is not Good. This information is provided by: Use case B5 or by alarms
Frequency	
Description	
Step	Action
1	OPC Client User opens UI of the device
2	User requests diagnostic data of the device
3	OPC client requests diagnostic data of the device from the OPC server
4	OPC server gets diagnostic data from the DTM
5	OPC server sends the diagnostic data to the client, displaying the data on the UI
Exceptions	
Requirements	
Notes	According to FDT2.0 the expectation is, that the DTM provides the parameters for device specific diagnostic within one group of parameters.
Open Issues	

B.7 Read parameters

B.7.1 Use case: Read offline data

Use case	Read offline parameters
Description	A software or user requests device parameter (the last set values) from a DTM. The OPC UA client collects this information by reading device parameter from the offline set provided according to the FDT OPC UA information model.
Stakeholder	
Preconditions	Device node is available in OPC UA Server and the Client already knows the ParameterIds of the Parameters to read (may have been retrieved according to use case B4.2).
Post conditions	Client knows the current value of device parameters
Actors	Engineer, Expert, Observer, Operator
Trigger	Whenever the data is needed
Frequency	
Description	
Step	Action
Preparation	The OPC Client selects the respective device node and the parameter set of the device.
1	OPC Client executes Read-Service (one or multiple times) with the list of identifiers for the respective device parameters (list of ParameterIds) and the list of AttributeIds for Value and (at least once) for DateType.
2	OPC Server retrieves the requested attributes (e.g. value) of the parameter which are specified in the Read-Request from the DTM (instance data set) and returns them to the client.
5	OPC Client displays the parameter value (or uses it otherwise).
Exceptions	Step 1, The specified parameter does not exist in the device.
Requirements	
Notes	
Open Issues	

B.7.2 Use case: Read online data

Use case	Read online parameters
Description	A software or user requests device parameter (the current values in the device) from a DTM. The OPC UA client collects this information by reading device parameter from the online set provided according to the FDT OPC UA information model.
Stakeholder	
Preconditions	Device node is available in OPC UA Server and the Client already knows the ParameterIds of the Parameters to read (may have been retrieved according to use case B4.2).
Post conditions	Client knows the current value of device parameters
Actors	Engineer, Expert, Observer, Operator
Trigger	Whenever the data is needed
Frequency	
Description	
Step	Action
Preparation	The OPC Client selects the respective offline device node
1	OPC Client selects the online device and the parameter set of the online device.
2	OPC Client executes Read-Service (one or multiple times) with the list of identifiers for the respective device parameters (list of ParameterIds) and the list of AttributeIds for Value and (at least once) for DateType.
3	OPC Server requests the requested attributes (e.g. value) of the parameter which are specified in the Read-Request from the DTM (instance data set).
4	The DTM retrieves the current values from the device and returns them to the server.
5	OPC Server returns the requested information to the client.
6	OPC Client displays the parameter value (or uses it otherwise).
Exceptions	Step 1, The specified parameter does not exist in the device.
Requirements	
Notes	
Open Issues	

B.8 Use case: Write device parameters

Use case	Writing Online Device Parameter
Description	The OPC client allows configuring or updating parameter values on the device. Writing is executed from an OPC client to an OPC server providing the FDT OPC UA information model.
Stakeholder	
Preconditions	Use Case B1 (List Topology)
Post conditions	OPC client can retrieve the updated device parameter values
Actors	Field Engineer, Device Expert, Maintenance Engineer
Trigger	Device Commissioning, Device Calibration, Device Configuration
Frequency	
Description	
Step	Action
2	User starts browse from Device Topology
3	OPC Client displays the topology to the user
4	User browse the parameters on the selected device
5	OPC Client displays available parameters on the device
6	User selects the device parameter that the value needs to be updated or changed
7	User changes the value of the selected device parameter based on its type
8	User commits the changes
9	OPC server puts the lock on the device
10	OPC server validates the specified parameter value
11	OPC server writes the updated value to the device
12	OPC server removes the lock on the device
13	OPC client displays the new value of the selected device parameter
Exceptions	Step 7, Value does not match the data type of the device parameter.
	Step 8, Device is in locked state.
	Step 10, Value exceeds the minimum and maximum limit specification of the device parameter.
Requirements	User has access permissions to the server
Notes	
Open Issues	

B.9 Use case: Audit trail

Use case	Audit Trail
Description	A central Audit Trail system logs audit events from FDT based tools. All auditable actions in the FDT tool produce OPC UA Audit Events and the central Audit Trail system subscribes for OPC UA Audit Events from the tool.
Used by	
Preconditions	Audit Events are supported by the OPC UA Server
Post conditions	Configuration changes are logged in the central Audit Trail.
Actors	Administrator for central Audit Trail system, Engineer, Expert
Trigger	Whenever device configuration actions must be logged in an audit trail
Frequency	
Description	
Step	Action
1	Define configuration changes that create an audit event in the OPC UA Server and FDT Frame
2	Activate Audit Events in the OPC UA Server
3	Configure central Audit Trail system to subscribe for audit events from the FDT OPC UA Server
4	Log configuration changes in the central Audit Trail system
5	View Audit Trail through a tool that comes with the central Audit Trail system
Exceptions	
Requirements	
Notes	
Open Issues	

Bibliography

[1] OPC 10000-100, OPC Unified Architecture – Part 100: Devices; Release 1.02; 2019-04-18;http://www.opcfoundation.org/UA/Part100/

_____

SOMMAIRE

AVANT-PROPOS 79
INTRODUCTION 81
0.1 Généralités 81
0.2 Présentation des outils FDT 81
0.3 Présentation de l'architecture unifiée OPC 81
0.4 Présentation de l'intégration des dispositifs OPC UA 82
1 Domaine d'application 84
2 Références normatives 84
3 Termes, définitions et termes abrégés 84
3.1 Termes et définitions 84
3.2 Termes abrégés 85
4 Conventions utilisées dans le présent document 85
4.1.1 Conventions pour les documents 85
4.1.2 Conventions pour les méthodes FDT 85
4.1.3 Conventions pour les descriptions de Nœuds 85
4.1.4 NodeIds et BrowseNames 88
4.1.5 Attributs communs 89
4.1.6 Notation graphique 91
5 Concept 92
5.1 Architecture du système 92
6 ObjectTypes OPC UA spécifiques à la FDT 93
6.1 Généralités 93
6.2 FdtDeviceType 93
6.3 FdtFunctionalGroupType 95
6.4 Interface IFdtDeviceHealthType 95
6.5 Interface IFdtSupportInfoType 95
6.5.1 Présentation 95
6.6 Types de documents 96
6.6.1 FdtDocumentType 96
6.6.2 FdtDocumentFile 97
6.6.3 FdtDocumentUrl 97
6.7 FdtProtocolType 97
6.8 FdtTransferServiceType 98
6.9 FdtIoSignalInfoType 98
7 EventTypes OPC UA 100
7.1 Présentation 100
7.2 FdtAuditEventType 100
7.3 FdtStartMethodEventType 100
7.4 FdtEndMethodEventType 101
7.5 FdtAuditWriteUpdateEventType 101
8 VariableTypes OPC UA 101
8.1 FdtParameter 101
9 DataTypes OPC UA 103
9.1 DataRefType 103
9.2 FdtDeviceClassificationType 103
9.3 SemanticInfoType 104

9.4 Datatypes Énumération 104
9.4.1 AlarmType 104
9.4.2 ApplicationIdEnumeration 105
9.4.3 ClassificationDomainId 105
9.4.4 ClassificationId 106
9.4.5 DocumentClassification 108
9.4.6 FunctionExecutionResultState 108
9.4.7 IECDatatype 109
9.4.8 RangeType 110
9.4.9 SignalTypeEnumeration 110
9.4.10 SubstitutionType 110
9.4.11 SupportedTransfer 111
10 ReferenceType OPC UA – HasIOSignalRef 112
11 Mapping de DataTypes 112
11.1 Types de données primitives – Énumération DeviceHealth 112
11.2 Mapping avec types OPC DI 113
11.2.1 Type de dispositif 113
11.2.2 TopologyElementType 117
11.2.3 FunctionalGroupType 119
11.2.4 FunctionalGroup Identification 120
11.2.5 Données et méthodes de dispositif 121
11.2.6 Méthodes 122
11.2.7 Variable 125
11.2.8 Informations de prise en charge de dispositif 128
11.2.9 FdtProtocolType 129
12 Profils et Unités de Conformité 129
12.1 Unités de Conformité 129
12.2 Profils 130
12.2.1 Liste des profiles 130
12.2.2 Facettes serveur 130
12.2.3 Facettes Client 131
13 Espaces de noms 132
13.1 Métadonnées des espaces de noms 132
13.2 Gestion des espaces de noms OPC UA 133
Annexe A (normative) Espaces de noms et identificateurs FDT 134
Annexe B (informative) Cas d'utilisation 135
B.1 Généralités 135
B.2 Cas d'utilisation: Topologie de liste 135
B.3 Cas d'utilisation: Identifier un dispositif 136
B.4 Obtenir la liste des paramètres disponibles du dispositif 137
B.4.1 Cas d'utilisation: Parcourir les paramètres du dispositif 137
B.4.2 Cas d'utilisation: Obtenir les attributs d'un paramètre du dispositif 138
B.5 Cas d'utilisation: Obtenir l'état du dispositif 139
B.6 Cas d'utilisation: Obtenir le diagnostic du dispositif 140
B.7 Lire les paramètres 141
B.7.1 Cas d'utilisation: Lire des données hors ligne 141
B.7.2 Cas d'utilisation: Lire des données en ligne 142
B.8 Cas d'utilisation: Ecriture des paramètres du dispositif 143

B.9 Cas d'utilisation: Piste de vérification 144
Bibliographie 145

Figure 1 – Exemple de Dispositifs OPC UA 83
Figure 2 – Notation du modèle d'information OPC UA 91
Figure 3 – Architecture du système conformément à l'IEC 62453-42 93
Figure 4 – Vue d'ensemble du FdtDeviceType 94
Figure 5 – Vue d'ensemble du FdtProtocolType 97
Figure 6 – Vue d'ensemble du FdtTransferServiceType 98
Figure 7 – Vue d'ensemble du FdtIoSignalInfoType 99
Figure 8 – Vue d'ensemble du type Évènement d'audit 100
Figure 9 – Exemple de sources d'informations DeviceType 113
Figure 10 – Exemple de sources d'informations TopologyType 118
Figure 11 – Exemple de mapping des données et des informations de fonctions 122
Figure 12 – Exemple de source d'informations de fonction 123
Figure 13 – Exemple de source d'informations de fonction statique 124

Tableau 1 – Exemples de DataTypes 86
Tableau 2 – Exemple de définition des types 87
Tableau 3 – Exemples d'autres caractéristiques 87
Tableau 4 – <Quelques> références types supplémentaires 87
Tableau 5 – <Quelques> sous-composantes supplémentaires 88
Tableau 6 – <Quelques> valeurs d'attributs de type pour Nœuds enfants 88
Tableau 7 – Attributs de Nœud communs 89
Tableau 8 – Attributs d'Objet communs 90
Tableau 9 – Attributs de Variable communs 90
Tableau 10 – Attributs de VariableType communs 90
Tableau 11 – Attributs de Méthode communs 91
Tableau 12 – Définition de FdtDeviceType 94
Tableau 13 – Définition de FdtFunctionalGroupType 95
Tableau 14 – Définition de IFdtDeviceHealthType 95
Tableau 15 – Définition de IFdtSupportInfoType 96
Tableau 16 – Sous-composantes supplémentaires du IFdtSupportInfoType 96
Tableau 17 – Définition de FdtDocumentType 96
Tableau 18 – Définition de FdtDocumentFile 97
Tableau 19 – Définition de FdtDocumentUrl 97
Tableau 20 – Définition de FdtProtocolType 98
Tableau 21 – Définition de FdtTransferServiceType 98
Tableau 22 – Définition de FdtIoSignalInfoType 99
Tableau 23 – Définition de FdtAuditEventType 100
Tableau 24 – Définition de FdtStartMethodEventType 101
Tableau 25 – Définition de FdtEndMethodEventType 101
Tableau 26 – Définition de FdtAuditWriteUpdateEventType 101

Tableau 27 – Définition de FdtParameter 102
Tableau 28 – Structure du DataRefType 103
Tableau 29 – Définition de DataRefType 103
Tableau 30 – Structure du FdtDeviceClassificationType 103
Tableau 31 – Définition de FdtDeviceClassificationType 104
Tableau 32 – Structure du SemanticInfoType 104
Tableau 33 – Définition de SemanticInfoType 104
Tableau 34 – Éléments d'AlarmType 104
Tableau 35 – Définition d'AlarmType 105
Tableau 36 – Éléments d'ApplicationIdEnumeration 105
Tableau 37 – Définition d'ApplicationIdEnumeration 105
Tableau 38 – Éléments de ClassificationDomainId 106
Tableau 39 – Définition de ClassificationDomainId 106
Tableau 40 – Éléments de ClassificationId 106
Tableau 41 – Définition de ClassificationId 108
Tableau 42 – Éléments de DocumentClassification 108
Tableau 43 – Définition de DocumentClassification 108
Tableau 44 – Éléments de FunctionExecutionResultState 108
Tableau 45 – Définition de FunctionExecutionResultState 109
Tableau 46 – Éléments d'IECDatatype 109
Tableau 47 – Définition de IECDatatype 109
Tableau 48 – Éléments de RangeType 110
Tableau 49 – Définition de RangeType 110
Tableau 50 – Éléments de SignalTypeEnumeration 110
Tableau 51 – Définition de SignalTypeEnumeration 110
Tableau 52 – Éléments de SubstitutionType 111
Tableau 53 – Définition de SubstitutionType 111
Tableau 54 – Éléments de SupportedTransfer 111
Tableau 55 – Définition de SupportedTransfer 111
Tableau 56 – Définition de HasIOSignalRef 112
Tableau 57 – Mapping pour DeviceHealthEnumeration 112
Tableau 58 – Mapping du DeviceType 114
Tableau 59 – Mapping des informations de dispositif 115
Tableau 60 – Mapping des paramètres du dispositif hors ligne 116
Tableau 61 – Mapping des paramètres du dispositif en ligne 117
Tableau 62 – Mapping du TopologyElementType 119
Tableau 63 – Mapping du FunctionalGroupType 120
Tableau 64 – Mapping pour FunctionalGroup Identification 120
Tableau 65 – Mapping des informations du nœud Méthode 123
Tableau 66 – Mapping des informations du nœud Méthode pour la fonction statique 124
Tableau 67 – Mapping des TransferServices 125
Tableau 68 – Mapping des éléments de données FDT 125
Tableau 69 – Mapping du FdtParameter 126

Tableau 70 – Mapping des types de données simples 127
Tableau 71 – Mapping de l'image de type de dispositif 128
Tableau 72 – Mapping de la ProtocolSupport 128
Tableau 73 – Mapping des informations du nœud FdtIoSignalInfoType 129
Tableau 74 – Mapping des informations du nœud FdtProtocolType 129
Tableau 75 – Unités de Conformité pour outils FDT 130
Tableau 76 – URI de profils pour outils FDT 130
Tableau 77 – Profil Serveur Base FDT 131
Tableau 78 – Facette Serveur Général FDT 131
Tableau 79 – Facette Client Général FDT 132
Tableau 80 – Objet NamespaceMetadata pour le présent document 132
Tableau 81 – Espaces de noms utilisés dans un Serveur FDT 133
Tableau 82 – Espaces de noms utilisés dans le présent document 133

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

SPÉCIFICATION DES INTERFACES DES OUTILS DES DISPOSITIFS DE TERRAIN (FDT) –

Partie 71: Modèle d'information de l'OPC UA pour outils FDT

AVANT-PROPOS

1) La Commission Electrotechnique Internationale (IEC) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de l'IEC). L'IEC a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. À cet effet, l'IEC – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de l'IEC"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec l'IEC, participent également aux travaux. L'IEC collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de l'IEC concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de l'IEC intéressés sont représentés dans chaque comité d'études.
3) Les Publications de l'IEC se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de l'IEC. Tous les efforts raisonnables sont entrepris afin que l'IEC s'assure de l'exactitude du contenu technique de ses publications; l'IEC ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de l'IEC s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de l'IEC dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de l'IEC et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) L'IEC elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de l'IEC. L'IEC n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à l'IEC, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de l'IEC, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de l'IEC ou de toute autre Publication de l'IEC, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'IEC attire l'attention sur le fait que la mise en application du présent document peut entraîner l'utilisation d'un ou de plusieurs brevets. L'IEC ne prend pas position quant à la preuve, à la validité et à l'applicabilité de tout droit de brevet revendiqué à cet égard. À la date de publication du présent document, l'IEC n'a pas reçu notification qu'un ou plusieurs brevets pouvaient être nécessaires à sa mise en application. Toutefois, il y a lieu d'avertir les responsables de la mise en application du présent document que des informations plus récentes sont susceptibles de figurer dans la base de données de brevets, disponible à l'adresse https://patents.iec.ch. L'IEC ne saurait être tenue pour responsable de ne pas avoir identifié tout ou partie de tels droits de propriété.

L'IEC 62453-71 a été établie par le sous-comité 65E: Les dispositifs et leur intégration dans les systèmes de l'entreprise, du comité d'études 65 de l'IEC: Mesure, commande et automation dans les processus industriels. Il s'agit d'une Norme internationale.

Le texte de cette Norme internationale est issu des documents suivants:

Projet	Rapport de vote
65E/806/CDV	65E/897A/RVC

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à son approbation.

La langue employée pour l'élaboration de cette Norme internationale est l'anglais.

Ce document a été rédigé selon les Directives ISO/IEC, Partie 2, il a été développé selon les Directives ISO/IEC, Partie 1 et les Directives ISO/IEC, Supplément IEC, disponibles sous www.iec.ch/members_experts/refdocs. Les principaux types de documents développés par l'IEC sont décrits plus en détail sous www.iec.ch/standardsdev/publications.

Une liste de toutes les parties de la série IEC 62453, publiées sous le titre général *Spécification des interfaces des outils des dispositifs de terrain (FDT)*, se trouve sur le site Web de l'IEC.

Le comité a décidé que le contenu de ce document ne sera pas modifié avant la date de stabilité indiquée sur le site web de l'IEC sous webstore.iec.ch dans les données relatives au document recherché. À cette date, le document sera

- reconduit,
- supprimé, ou
- révisé.

IMPORTANT – Le logo "colour inside" qui se trouve sur la page de couverture de ce document indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer cette publication en utilisant une imprimante couleur.

INTRODUCTION

0.1 Généralités

La nouvelle Architecture unifiée de l'OPC (OPC UA) unifie les normes existantes et les met à la pointe de la technologie en utilisant une architecture orientée services (SOA). La technologie indépendante de la plateforme permet le déploiement de l'OPC UA au-delà des applications OPC actuelles fonctionnant uniquement sur des systèmes PC basés sur Windows. L'OPC UA peut également fonctionner sur des systèmes embarqués, ainsi que sur des systèmes d'entreprise basés sur Linux/UNIX. Les informations fournies peuvent être modélisées de manière générique et, par conséquent, des modèles d'information arbitraires peuvent être fournis en utilisant l'OPC UA.

Les outils FDT normalisent l'interface de communication et de configuration entre tous les dispositifs de terrain et les systèmes hôtes. Les outils FDT fournissent un environnement commun pour accéder aux fonctions les plus sophistiquées des dispositifs. Tout dispositif peut être configuré, exploité et entretenu par l'intermédiaire de l'interface utilisateur normalisée, indépendamment du fournisseur, du type ou du protocole de communication.

Le présent document spécifie une synergie des deux approches, offrant ainsi un accès facile et normalisé par des interfaces OPC UA à la connaissance des dispositifs basée sur les outils FDT.

0.2 Présentation des outils FDT

Les outils FDT constituent une technologie prenant en charge l'échange de données entre dispositifs de terrain et systèmes d'automatisation. Cette technologie est basée sur une spécification d'interface normalisée conformément à l'IEC 62453. La spécification définit deux concepts principaux: le gestionnaire de type d'équipements (DTM) et l'application-cadre. Un gestionnaire de type d'équipements est un composant logiciel spécifique à un type de dispositifs de terrain. Une application-cadre est un environnement logiciel (faisant partie du système d'automatisation) pour l'intégration des gestionnaires de type d'équipements. Dans une application-cadre, chaque gestionnaire de type d'équipements fournit des données et des services spécifiques au dispositif de terrain concerné. Comme la technologie repose sur un ensemble d'interfaces normalisées, chaque gestionnaire de type d'équipements peut être intégré dans chaque application-cadre. Grâce aux outils FDT, il est possible d'intégrer des dispositifs de communication, des dispositifs d'infrastructure de communication (par exemple des passerelles) et des dispositifs de terrain, en fonction de leurs protocoles de communication. Les différents protocoles de communication sont pris en charge par des spécifications de protocoles de communication supplémentaires (par exemple pour PROFINET, PROFIBUS, Ethernet IP, TCP, HART et FF) ou par l'intégration de protocoles spécifiques à un fabricant.

0.3 Présentation de l'architecture unifiée OPC

Le principal cas d'utilisation des normes OPC est l'échange de données en ligne entre des dispositifs et des systèmes IHM ou SCADA utilisant une fonctionnalité d'accès aux données. Dans ce cas d'utilisation, les données d'un dispositif sont fournies par un serveur d'OPC et sont consommées par un client OPC intégré dans le système HMI ou SCADA. L'OPC DA fournit la fonctionnalité pour naviguer à travers des espaces de noms hiérarchiques contenant des éléments de données, ainsi que pour lire, écrire et surveiller ces éléments pour des modifications de données. Les spécifications OPC Classic sont basées sur la technologie de COM/DCOM de Microsoft pour la communication entre des composants logiciels provenant de différents fournisseurs. Par conséquent, le serveur et les clients OPC Classic sont limités aux systèmes d'automation basés sur les systèmes d'exploitation Windows.

L'architecture OPC UA comporte toutes les caractéristiques des spécifications de classe OPC comme OPC DA, A&E et HDA, mais elle définit les mécanismes de communication indépendants des plateformes et des capacités de modélisation génériques, orientées objet et extensibles pour les informations qu'un système souhaite présenter.

La partie communication de réseau de l'OPC UA définit différents mécanismes optimisés pour différents cas d'utilisation. La première version de l'OPC UA définit un protocole binaire optimisé pour la communication intranet haute performance, ainsi qu'un mapping avec des normes internet reconnues comme les services web. Le modèle de communication abstrait ne dépend pas d'un mapping de protocole spécifique et permet l'ajout de nouveaux protocoles dans le futur. Les caractéristiques telles que la sécurité, le contrôle d'accès et la fiabilité sont directement intégrées dans les mécanismes de transport. Sur la base de l'indépendance des protocoles vis-à-vis de toute plateforme, les Serveurs et Clients OPC UA peuvent être directement intégrés dans les dispositifs et les contrôleurs.

Le modèle d'information de l'OPC UA fournit un moyen normalisé à des Serveurs pour présenter des Objets à des Clients. Les Objets au sens de l'OPC UA se composent d'autres Objets, de Variables et de Méthodes. L'OPC UA permet aussi l'expression de relations à d'autres Objets.

L'ensemble des Objets et des relations connexes qu'un Serveur de l'OPC UA met à la disposition des Clients est appelé son AddressSpace (espace d'adresses). Les éléments du modèle d'Objets OPC UA sont représentés dans l'AddressSpace sous la forme d'un ensemble de Nodes (nœuds) décrits par des Attributs et interconnectés par des Références. L'OPC UA définit huit classes de Nœuds pour représenter les composants de l'AddressSpace. Les classes sont Object (objet), Variable, Method (méthode), ObjectType (type d'objet), VariableType (type de variable), DataType (type de données), ReferenceType (type de référence) et View (vue). Chaque NodeClass (classe de nœuds) a un ensemble défini d'Attributs.

La présente spécification utilise presque toutes les NodeClasses de l'OPC UA: Objets et Variables.

Les Objets sont utilisés pour représenter les composants d'un système. Un Objet est associé à un ObjectType correspondant qui fournit des définitions pour l'Objet en question.

Les Variables sont utilisées pour représenter des valeurs. Deux catégories de Variables sont définies: les Propriétés et les DataVariables.

Les Propriétés sont des caractéristiques d'Objets, de DataVariables et d'autres Nœuds, définis par le Serveur. Les Propriétés ne sont pas autorisées à avoir des Propriétés qui sont définies pour elles. La Propriété de révision d'un DeviceType constitue un exemple de Propriétés d'Objets.

Les DataVariables représentent le contenu d'un Objet. Les DataVariables sont susceptibles d'avoir des composants DataVariables. Elles sont typiquement utilisées par les Serveurs pour présenter des éléments individuels de matrices et de structures. La présente spécification utilise les DataVariables pour représenter des données comme les variables de processus fournies par un dispositif.

0.4 Présentation de l'intégration des dispositifs OPC UA

La spécification "Intégration des dispositifs OPC UA" est une extension de la série globale de spécifications de l'OPC UA (architecture unifiée OPC) et définit le modèle d'information associé aux dispositifs. Le modèle vise à fournir une vue unifiée des dispositifs et ce, indépendamment des protocoles de dispositifs sous-jacents. Les outils FDT traitent des dispositifs physiques ou logiques et le modèle d'information de l'IEC 62541-100 est donc utilisé comme base pour le modèle d'information FDT.

Le modèle d'information des dispositifs spécifie différents ObjectTypes et procédures utilisés pour représenter les dispositifs et les composants connexes comme l'infrastructure de communication dans un espace d'adressage OPC UA. Les principaux cas d'utilisation sont la configuration et le diagnostic des dispositifs, mais ils permettent à tout type d'application d'accéder aux informations relatives aux dispositifs de manière générale et normalisée. Les exemples suivants illustrent les concepts utilisés dans la présente spécification. Voir Dispositifs OPC-UA pour la définition complète du modèle d'information des dispositifs.

La Figure 1 représente un exemple d'un régulateur de température représenté comme un Objet Dispositif. Le composant ParameterSet contient toutes les Variables décrivant le Dispositif. Le composant MethodSet contient toutes les Méthodes fournies par le Dispositif. Ces deux composants sont hérités du TopologyElementType, qui est le type d'Objet racine de la hiérarchie des types d'Objets dispositifs. Les Objets de type FunctionalGroupType sont utilisés pour regrouper les Paramètres et Méthodes du Dispositif en groupes logiques. Le FunctionalGroupType et le concept de groupage sont définis dans Dispositifs UA, mais les groupes sont spécifiques au type de dispositif, c'est-à-dire que les groupes ProcessData et Configuration sont définis par le TemperatureControllerType dans cet exemple.

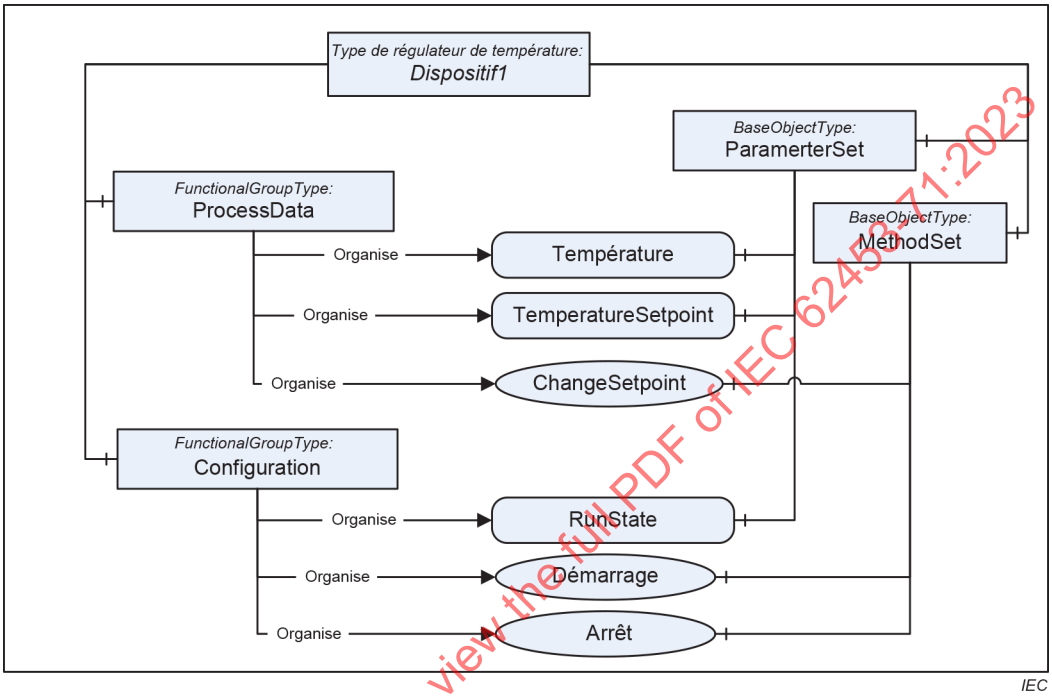


Figure 1 – Exemple de Dispositifs OPC UA

Les cas d'utilisation de l'Annexe B représentent l'utilisation du modèle d'information. Il n'est pas nécessaire que tous les Objets nécessaires soient réalisés dans un Serveur OPC UA concret.

SPÉCIFICATION DES INTERFACES DES OUTILS DES DISPOSITIFS DE TERRAIN (FDT) –

Partie 71: Modèle d'information de l'OPC UA pour outils FDT

1 Domaine d'application

La présente partie de l'IEC 62453 spécifie un modèle d'information de l'OPC UA pour représenter les informations des dispositifs, basé sur l'intégration du dispositif définie pour un outil FDT.

2 Références normatives

Les documents suivants sont cités dans le texte de sorte qu'ils constituent, pour tout ou partie de leur contenu, des exigences du présent document. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

IEC 62453-1:2023, *Spécification des interfaces des outils des dispositifs de terrain (FDT) – Partie 1: Vue d'ensemble et guide*

IEC 62453-2:2022, *Spécification des interfaces des outils des dispositifs de terrain (FDT) – Partie 2: Concepts et description détaillée*

IEC 62541-3:2020, *Architecture unifiée OPC – Partie 3: Modèle d'espace d'adressage*

IEC 62541-5:2020, *Architecture unifiée OPC – Partie 5: Modèle d'informations*

IEC 62541-6, *Architecture unifiée OPC – Partie 6: Mappings*

IEC 62541-7, *Architecture unifiée OPC – Partie 7: Profils*

IEC 62541-8, *Architecture unifiée OPC – Partie 8: Accès aux données*

IEC 62541-100:2015, *Architecture unifiée OPC – Partie 100: Interface d'appareils*

3 Termes, définitions et termes abrégés

3.1 Termes et définitions

Pour les besoins du présent document, les termes et définitions de l'IEC 62453-1, de l'IEC 62453-2, ainsi que de l'IEC 62451-100 s'appliquent.

L'ISO et l'IEC tiennent à jour des bases de données terminologiques destinées à être utilisées en normalisation, consultables aux adresses suivantes:

- IEC Electropedia: disponible à l'adresse https://www.electropedia.org/
- ISO Online browsing platform: disponible à l'adresse https://www.iso.org/obp

3.2 Termes abrégés

Pour les besoins du présent document, les abréviations de l'IEC 62453-1, de l'IEC 62453-2, ainsi que les suivants s'appliquent.

A&E Alarms & Events (alarmes et événements)
DA Data Access (accès aux données)
DTM Device Type Manager (gestionnaire de type d'équipements)
FDT Field Device Technology (technologie des dispositifs de terrain)
HDA Historical Data Access (accès aux données historiques)
IHM Interface Homme-Machine
IEC International Electrotechnical Commission (Commission Electrotechnique Internationale)
MES Manufacturing Execution System (système d'exécution de fabrication)
UA Unified Architecture (architecture unifiée)
XML Extensible Markup Language (langage de balisage extensible)

4 Conventions utilisées dans le présent document

4.1.1 Conventions pour les documents

Tout au long du présent document, certaines conventions documentaires sont utilisées.

Le format italique est utilisé pour mettre en évidence un terme ou une définition qui figure à l'Article 3 dans le présent document ou dans l'un des documents OPC UA de référence.

Le format italique est également utilisé pour mettre en évidence le nom d'un paramètre d'entrée ou de sortie de service, ou le nom d'une structure ou d'un élément de structure habituellement défini dans les tableaux.

4.1.2 Conventions pour les méthodes FDT

Les outils FDT définissent des méthodes synchrones et asynchrones. Les méthodes asynchrones sont mises en œuvres avec un ensemble de méthodes. Par exemple, une méthode asynchrone <MethodName> est mise en œuvre avec Begin<MethodName>(), Cancel<MethodName>() et End<MethodName>().

Dans le présent document, les méthodes asynchrones sont indiquées par les caractères "<>" devant le nom (par exemple <>MethodName).

4.1.3 Conventions pour les descriptions de Nœuds

4.1.3.1 Généralités

Les définitions de Nœuds sont spécifiées en utilisant des tableaux (voir Tableau 2).

Les *Attributs* sont définis en fournissant le nom d'*Attribut* et une valeur, ou une description de la valeur.

Les *Références* sont définies en fournissant le nom du *ReferenceType*, le *BrowseName* du *TargetNode* et sa *NodeClass*.

- Si le *TargetNode* est un composant du *Nœud* défini dans le tableau, les *Attributs* du *Nœud* composé sont définis dans la même ligne du tableau.

- Le *DataType* est uniquement spécifié pour les *Variables*; "[<nombre>]" indique une matrice unidimensionnelle, alors que pour les matrices multidimensionnelles, l'expression est répétée pour chaque dimension (par exemple [2][3] pour une matrice à deux dimensions). Pour toutes les matrices, *ArrayDimensions* est établi comme identifié par des valeurs <nombre>. Si aucun <nombre> n'est défini, la dimension correspondante est définie sur 0, indiquant une taille inconnue. Si aucun nombre n'est fourni, *ArrayDimensions* peut être omis. L'absence de crochets identifie un *DataType* scalaire et le *ValueRank* est défini sur la valeur correspondante (voir IEC 62541-3). De plus, *ArrayDimensions* est défini sur null ou est omis. Si elle peut être Any (tout) ou *ScalarOrOneDimension*, la valeur est insérée dans "{<valeur>}" et ainsi "{Any}" ou "{*ScalarOrOneDimension*}" et le *ValueRank* sont définis sur la valeur correspondante (voir IEC 62541-3) et *ArrayDimensions* est défini sur null ou est omis. Des exemples sont donnés dans le Tableau 1.

Tableau 1 – Exemples de DataTypes

Notation	**DataType**	**Value Rank**	**ArrayDimensions**	**Description**
0:Int32	0:Int32	-1	omis ou nul	Valeur Int32 scalaire
0:Int32[]	0:Int32	1	omis ou {0}	Matrice unidimensionnelle de Int32 de taille inconnue
0:Int32[][]	0:Int32	2	omis ou {0,0}	Matrice bidimensionnelle de Int32 de taille inconnue pour les deux dimensions
0:Int32[3][]	0:Int32	2	{3,0}	Matrice bidimensionnelle de Int32 de taille 3 pour la première dimension et de taille inconnue pour la seconde dimension
0:Int32[5][3]	0:Int32	2	{5,3}	Matrice bidimensionnelle de Int32 de taille 5 pour la première dimension et de taille 3 pour la seconde dimension
0:Int32{Any}	0:Int32	-2	omis ou nul	Int32 dans le cas d'un scalaire ou d'une matrice d'une quelconque dimension
0:Int32{ScalarOrOne Dimension}	0:Int32	-3	omis ou nul	Int32 dans le cas d'une matrice unidimensionnelle ou d'un scalaire

- La TypeDefinition est spécifiée pour les *Objets* et les *Variables*.
- La colonne TypeDefinition spécifie un nom symbolique pour un *NodeId*, c'est-à-dire les points de *Nœud* spécifiés avec une *Référence HasTypeDefinition* au *Nœud* correspondant.
- La *ModellingRule* de la composante référencée est fournie par la spécification du nom symbolique de la règle dans la colonne *ModellingRule*. Dans l'*AddressSpace*, le *Nœud* doit utiliser une *Référence HasModellingRule* pour pointer vers l'*Objet ModellingRule* correspondant.

Si le *NodeId* d'un *DataType* est fourni, le nom symbolique du *Nœud* représentant le *DataType* doit être utilisé.

Noter que si un nom symbolique d'un espace de noms différent est utilisé, il est préfixé par le *NamespaceIndex* (voir 4.1.4.2).

Les *Nœuds* de toutes les autres *NodeClasses* ne peuvent pas être définis dans le même tableau; par conséquent, seuls le *ReferenceType* utilisé, sa *NodeClass* et son *BrowseName* sont spécifiés. Une référence à une autre partie du présent document pointe vers leur définition.

Le Tableau 2 représente le tableau correspondant. Si aucune composante n'est fournie, les colonnes DataType, TypeDefinition et Autres peuvent être omises et seule la colonne Commentaires est introduite pour pointer vers la définition du *Nœud*.

Tableau 2 – Exemple de définition des types

Attribut	**Valeur**				
Nom de l'attribut	Valeur d'attribut. S'il s'agit d'un attribut facultatif qui n'est pas défini, "--" est utilisé.				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Nom du ReferenceType	NodeClass du TargetNode	BrowseName du TargetNode	DataType du Nœud référencé, applicable seulement aux Variables	TypeDefinition du Nœud référencé, applicable seulement aux Variables et Objets	Caractéristiques supplémentaires du TargetNode, comme ModellingRule ou AccessLevel
NOTE Notes référençant les notes concernant le contenu du tableau.					

Les composantes de *Nœuds* peuvent être complexes, c'est-à-dire contenir elles-mêmes des composantes. Les *TypeDefinition*, *NodeClass* et *DataType* peuvent résulter des définitions de type, et le nom symbolique peut être créé comme défini en 4.1.5.1. Par conséquent, celles qui contiennent des composantes ne sont pas spécifiées de manière explicite; elles sont implicitement spécifiées par les définitions de type.

La colonne Autres définit des caractéristiques supplémentaires du Nœud. Des exemples de caractéristiques qui peuvent figurer dans cette colonne sont représentés dans le Tableau 3.

Tableau 3 – Exemples d'autres caractéristiques

Name	**Abréviation**	**Description**
0:Mandatory	M	Le *Nœud* a la *Mandatory ModellingRule*.
0:Optional	O	Le *Nœud* a la *Optional ModellingRule*.
0:MandatoryPlaceholder	MP	Le *Nœud* a la *MandatoryPlaceholder ModellingRule*.
0:OptionalPlaceholder	OP	Le *Nœud* a la *OptionalPlaceholder ModellingRule*.
ReadOnly	RO	Le bit *CurrentRead* du Nœud *AccessLevel* est défini, mais pas le bit *CurrentWrite*.
ReadWrite	RW	Le bit *CurrentRead* et le bit *CurrentWrite* du *Nœud AccessLevel* sont définis.
WriteOnly	WO	Le bit *CurrentWrite* du Nœud *AccessLevel* est défini, mais pas le bit *CurrentRead*.

Si plusieurs caractéristiques sont définies, elles sont séparées par des virgules. Le nom ou son abréviation peuvent être utilisés.

4.1.3.2 Références supplémentaires

Pour fournir des informations sur des *Références* supplémentaires, le format représenté dans le Tableau 4 est utilisé.

Tableau 4 – <Quelques> références types supplémentaires

SourceBrowsePath	**Type de référence**	**Directe**	**TargetBrowsePath**
SourceBrowsePath est toujours relatif au *TypeDefinition*. Les éléments multiples sont définis comme des lignes distinctes d'un tableau imbriqué.	Nom du *ReferenceType*	True = *Référence* directe.	TargetBrowsePath pointe vers un autre *Nœud*, qui peut être une instance connue ou un *TypeDefinition*. *BrowsePaths* peut également être utilisé dans ce cas et il est soit relatif au *TypeDefinition* soit absolu. S'il est absolu, il est nécessaire que la première entrée fasse référence à un type ou à une instance bien connue, identifiée de manière unique dans un espace de noms par le *BrowseName*.

Les *Références* peuvent concerner n'importe quel autre *Nœud*.

4.1.3.3 Sous-composantes supplémentaires

Pour fournir des informations sur des sous-composantes, le format représenté dans le Tableau 5 est utilisé.

Tableau 5 – <Quelques> sous-composantes supplémentaires

BrowsePath	Références	NodeClass	BrowseName	DataType	TypeDefinition	Autres
BrowsePath est toujours relatif au *TypeDefinition*. Les éléments multiples sont définis comme des lignes distinctes d'un tableau imbriqué.	NOTE Idem que pour le Tableau 2					

4.1.3.4 Valeurs d'attributs supplémentaires

Le tableau de définitions des types contient des colonnes spécifiant les valeurs des *Attributs* de *Nœud* exigés pour *InstanceDeclarations*. Pour fournir des informations sur des *Attributs* supplémentaires, le format représenté dans le Tableau 6 est utilisé.

Tableau 6 – <Quelques> valeurs d'attributs de type pour Nœuds enfants

BrowsePath	Attribut <Attribute name>
BrowsePath est toujours relatif au *TypeDefinition*. Les éléments multiples sont définis comme des lignes distinctes d'un tableau imbriqué.	Les Valeurs des Attributs sont converties en texte en adaptant les règles de codage JSON réversible définies dans l'IEC 52541-6. Si le codage JSON d'une valeur est une chaîne JSON ou un nombre JSON, cette valeur est alors saisie dans le champ de valeur. Les guillemets ne sont pas inclus. Si le DataType comprend un NamespaceIndex (QualifiedNames, NodeIds ou ExpandedNodeIds), la notation utilisée pour BrowseNames est utilisée. Si la valeur est une Énumération, le nom de la valeur de l'énumération est saisi. Si la valeur est une structure, une séquence de paires de noms et de valeurs est saisie. Chaque paire est suivie d'un saut de ligne. Le nom est suivi de deux points. Les noms sont les noms des champs dans la DataTypeDefinition. Si la valeur est une matrice de non-structures, une séquence de valeurs est saisie, chaque valeur étant suivie d'un saut de ligne. Si la valeur est une matrice de structures ou une structure dont les champs sont des matrices ou des structures imbriquées, la matrice JSON complète ou l'objet JSON est saisi. Les guillemets ne sont pas inclus.

Il peut y avoir plusieurs colonnes pour définir plusieurs *Attributs*.

4.1.4 NodeIds et BrowseNames

4.1.4.1 NodeIds

Les *NodeIds* de tous les *Nœuds* décrits dans le présent document sont uniquement des noms symboliques. L'Annexe A définit les *NodeIds* réels.

Le nom symbolique de chaque *Nœud* défini dans le présent document est son *BrowseName* ou, lorsqu'il fait partie d'un autre *Nœud*, le *BrowseName* de l'autre *Nœud*, un "." suivi de son propre *BrowseName*. Dans ce cas, "fait partie de" signifie que l'ensemble possède une *Référence HasProperty* ou *HasComponent* à cette partie. Sachant que tous les *Nœuds* ne faisant pas partie d'un autre *Nœud* ont un nom unique dans le présent document, le nom symbolique est unique.

Le *NamespaceUn* pour tous les *NodeIds* définis dans le présent document est défini à l'Annexe A. Le *NamespaceIndex* pour ce *NamespaceUri* est spécifique à un fournisseur et dépend de la position du *NamespaceUri* dans le tableau d'espaces de noms du serveur.

NOTE Le présent document définit non seulement des *Nœuds* concrets, mais exige également que certains *Nœuds* soient créés, par exemple un nœud pour chaque *Session* exécutée sur le *Serveur*. Les *NodeIds* de ces *Nœuds* sont spécifiques au *Serveur*, y compris le Namespace. Toutefois, le *NamespaceIndex* de ces *Nœuds* ne peut pas être le *NamespaceIndex* utilisé pour les *Nœuds* définis dans le présent document, car ils ne sont pas définis par le présent document, mais générés par le *Serveur*.

4.1.4.2 BrowseNames

La partie texte des *BrowseNames* pour tous les *Nœuds* définis dans le présent document est spécifiée dans les tableaux définissant les *Nœuds*. Le *NamespaceUn* pour tous les *BrowseNames* définis dans le présent document est défini en 13.2.

Pour les *InstanceDeclarations* de l'*Objet* et de la *Variable NodeClass* qui sont des paramètres fictifs (*ModellingRule OptionalPlaceholder* et *MandatoryPlaceholder*), le *BrowseName* et le *DisplayName* sont placés entre crochets (<>) comme recommandé dans l'IEC 62541-3. Si le *BrowseName* n'est pas défini par le présent document, un préfixe d'index d'espace de nom est ajouté au *BrowseName* (par exemple le préfixe "0" qui donne "0:EngineeringUnits" ou le préfixe "2" qui donne "2:DeviceRevision"). Ceci est généralement nécessaire si une *Propriété* d'une autre spécification est écrasée ou utilisée dans les types OPC UA définis dans le présent document. Le Tableau 82 fournit une liste des espaces de noms et de leurs index utilisés dans le présent document.

4.1.5 Attributs communs

4.1.5.1 Généralités

Les *Attributs* des *Nœuds*, leurs *DataTypes* et descriptions sont définis dans l'IEC 62541-3. Les Attributs non marqués comme facultatifs sont obligatoires et doivent être fournis par un *Serveur*. Les tableaux suivants définissent si la valeur de l'*Attribut* est définie par le présent document ou si elle est spécifique au serveur.

Pour tous les *Nœuds* spécifiés dans le présent document, les *Attributs* énumérés dans le Tableau 7 doivent être définis comme spécifié dans le tableau.

Tableau 7 – Attributs de Nœud communs

Attribut	Valeur
DisplayName	Le *DisplayName* est un *LocalizedText*. Chaque *Serveur* doit fournir le *DisplayName* identique au *BrowseName* du *Nœud* pour le *LocaleId* "en". La possibilité que le serveur fournisse ou non des noms traduits pour d'autres *LocaleIds* est spécifique au serveur.
Description	Description spécifique au serveur (facultatif).
NodeClass	Doit refléter la *NodeClass* du *Nœud*.
NodeId	Le *NodeId* est décrit par des *BrowseNames* comme défini en 4.1.4.1.
WriteMask	L'*Attribut WriteMask* peut être donné (facultatif). Si l'*Attribut WriteMask* est fourni, il doit définir tous les *Attributs* non spécifiques au serveur comme non inscriptibles. Par exemple, l'*Attribut Description* peut être défini comme inscriptible, car un *Serveur* peut fournir une description spécifique au serveur pour le *Nœud*. Le *NodeId* ne doit pas être inscriptible, car il est défini pour chaque *Nœud* dans le présent document.
UserWriteMask	L'*Attribut UserWriteMask* peut être donné (facultatif). Les mêmes règles que dans le cas de l'*Attribut WriteMask* s'appliquent.
RolePermissions	Des autorisations de rôle spécifiques à un serveur peuvent être données (facultatif).
UserRolePermissions	Des autorisations de rôle pour la Session en cours peuvent être données (facultatif). La valeur est spécifique au serveur et dépend de l'*Attribut RolePermissions* (s'il est fourni) et de la *Session* en cours.
AccessRestrictions	Des restrictions d'accès spécifiques à un serveur peuvent être données (facultatif).

4.1.5.2 Objets

Pour tous les *Objets* spécifiés dans le présent document, les *Attributs* énumérés dans le Tableau 8 doivent être définis comme spécifié dans le Tableau 8. Les définitions des *Attributs* peuvent être consultées dans l'IEC 62541-3.

Tableau 8 – Attributs d'Objet communs

Attribut	Valeur
EventNotifier	La possibilité d'utiliser le *Nœud* pour s'abonner à des *Évènements* ou non est spécifique au serveur.

4.1.5.3 Variables

Pour toutes les *Variables* spécifiées dans le présent document, les *Attributs* énumérés dans le Tableau 9 doivent être définis comme spécifié dans le tableau. Les définitions des *Attributs* peuvent être consultées dans l'IEC 62541-3.

Tableau 9 – Attributs de Variable communs

Attribut	Valeur
MinimumSamplingInterval	Un intervalle d'échantillonnage minimal spécifique au serveur est donné (facultatif).
AccessLevel	Le niveau d'accès pour les *Variables* utilisées dans les définitions de types est spécifique au serveur; pour toutes les autres *Variables* définies dans le présent document, le niveau d'accès doit permettre la lecture; d'autres réglages sont spécifiques au serveur.
UserAccessLevel	La valeur pour l'*Attribut UserAccessLevel* est spécifique au serveur. Il est présumé que toutes les *Variables* soient accessibles à au moins un utilisateur.
Valeur	Pour les *Variables* utilisées comme *InstanceDeclarations*, la valeur est spécifique au serveur; dans le cas contraire, elle doit représenter la valeur décrite dans le texte.
ArrayDimensions	Si le *ValueRank* n'identifie pas une matrice d'une dimension spécifique (c'est-à-dire *ValueRank* ≤ 0), *ArrayDimensions* peut être défini sur nul ou bien l'*Attribut* est absent. Ce comportement est spécifique au serveur. Si le *ValueRank* spécifie une matrice d'une dimension spécifique (c'est-à-dire *ValueRank* > 0), l'*Attribut ArrayDimensions* doit être spécifié dans le tableau définissant la *Variable*.
Historizing	La valeur pour l'*Attribut Historizing* est spécifique au serveur.
AccessLevelEx	Si l'*Attribut AccessLevelEx* est donné, il doit avoir les bits 8, 9 et 10 définis sur 0, ce qui signifie que les opérations de lecture et d'écriture sur une *Variable* individuelle sont atomiques et que les matrices peuvent être en partie écrites.

4.1.5.4 VariableTypes

Pour toutes les *VariableTypes* spécifiées dans le présent document, les *Attributs* énumérés dans le Tableau 10 doivent être définis comme spécifié dans le tableau. Les définitions des *Attributs* peuvent être consultées dans l'IEC 62541-3.

Tableau 10 – Attributs de VariableType communs

Attributs	Valeur
Valeur	Une valeur par défaut spécifique au serveur peut être donnée (facultatif).
ArrayDimensions	Si le *ValueRank* n'identifie pas une matrice d'une dimension spécifique (c'est-à-dire *ValueRank* ≤ 0), *ArrayDimensions* peut être défini sur nul ou bien l'*Attribut* est absent. Ce comportement est spécifique au serveur. Si le *ValueRank* spécifie une matrice d'une dimension spécifique (c'est-à-dire *ValueRank* > 0), l'*Attribut ArrayDimensions* doit être spécifié dans le tableau définissant le *VariableType*.

4.1.5.5 Méthodes

Pour toutes les *Méthodes* spécifiées dans le présent document, les *Attributs* énumérés dans le Tableau 11 doivent être définis comme spécifié dans le tableau. Les définitions des *Attributs* peuvent être consultées dans l'IEC 62541-3.

Tableau 11 – Attributs de Méthode communs

Attributs	Valeur
Executable	Toutes les *Méthodes* définies dans le présent document doivent être exécutables (*Attribut Executable* défini sur "True"), à moins d'être définies différemment dans la définition de la *Méthode*.
UserExecutable	La valeur pour l'*Attribut UserExecutable* est spécifique au serveur. Il est admis par hypothèse que toutes les *Méthodes* puissent être exécutées par au moins un utilisateur.

4.1.6 Notation graphique

4.1.6.1 Généralités

L'OPC UA définit une notation graphique d'un *AddressSpace* OPC UA. Il définit les symboles graphiques pour toutes les NodeClasses et la façon dont les différents types de *Références* entre les Nœuds peuvent être visualisés. La Figure 2 représente les symboles des NodeClasses utilisées dans la présente spécification. Les NodeClasses représentant des types ont toujours une ombre.

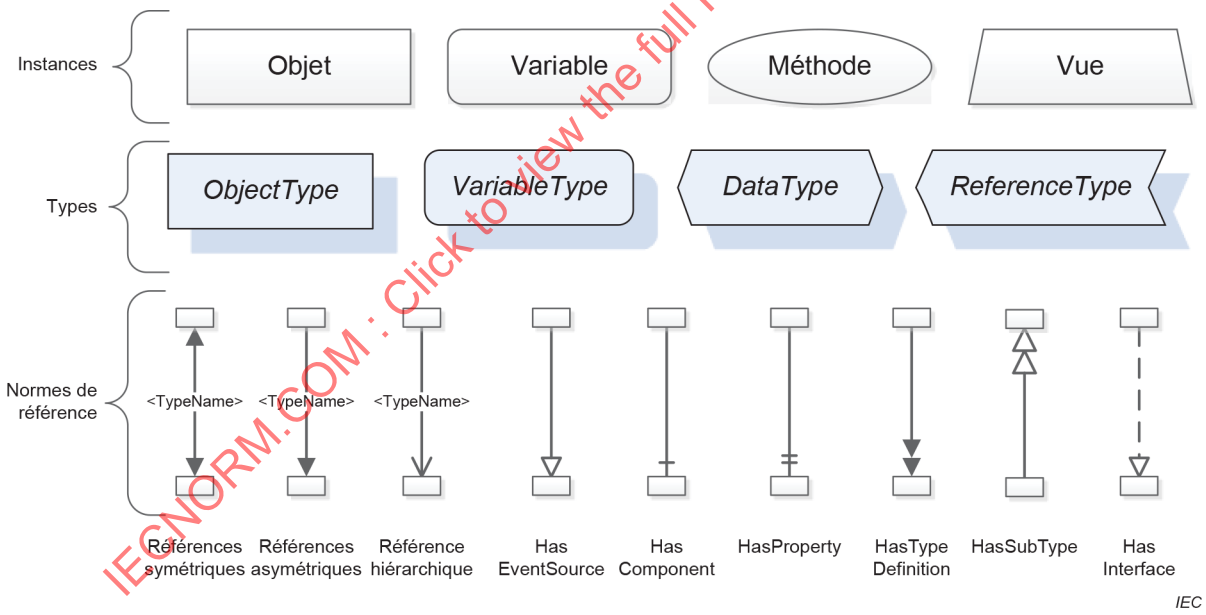


Figure 2 – Notation du modèle d'information OPC UA

Une description complète des différents types de Nœuds et de Références se trouve dans l'IEC 62541-3 et la structure de base est décrite dans l'IEC 62541-5.

La spécification OPC UA définit une très large gamme de fonctionnalités dans son modèle d'information de base. Il n'est pas exigé que tous les *Clients* ou *Serveurs* prennent en charge toutes les fonctionnalités dans les spécifications OPC UA. L'OPC UA inclut le concept de *Profils*, qui segmentent la fonctionnalité en unités certifiables pouvant être mises à l'essai. Cela permet de définir des sous-ensembles fonctionnels (qui sont censés être mis en œuvre) dans une spécification d'accompagnement. Les *Profiles* ne limitent pas la fonctionnalité, mais génèrent des exigences pour un ensemble minimal de fonctionnalités (voir IEC 62541-7).

4.1.6.2 Espaces de noms

L'OPC UA permet de combiner des informations provenant de nombreuses sources différentes en un seul *AddressSpace* cohérent. Les espaces de noms sont utilisés à cette fin en éliminant les conflits de noms et d'identifiants entre des informations provenant de sources différentes. Chaque Namespace de l'OPC UA a une chaîne globale unique appelée NamespaceUri qui identifie une autorité de nommage et un entier local unique appelé *NamespaceIndex*, qui est un index dans le tableau des *NamespaceUri* du *Serveur*. Le *NamespaceIndex* est unique seulement dans le contexte d'une *Session* entre un *Client* OPC UA et un *Serveur* OPC UA – le *NamespaceIndex* peut changer entre *Sessions* et toujours identifier le même élément même si l'emplacement du NamespaceUri dans le tableau a changé. Les *Services* définis pour l'OPC UA utilisent le *NamespaceIndex* pour spécifier le Namespace pour des valeurs qualifiées.

Il existe deux types de valeurs structurées dans l'OPC UA qui sont qualifiées par des *NamespaceIndexes*: NodeIds et *QualifiedNames*. Les NodeIds sont des identificateurs locaux uniques (et parfois globaux uniques) pour les *Nœuds*. Le même *NodeId* global unique peut être utilisé comme identificateur d'un nœud dans de nombreux *Serveurs*, les données d'instance du nœud peuvent varier, mais sa signification sémantique est la même indépendamment du *Serveur* dans lequel il figure. Cela signifie que les *Clients* peuvent avoir une connaissance intégrée de la signification des données dans ces *Nœuds*. Les *Modèles d'information* OPC UA définissent généralement des *NodeIds* globaux uniques pour les *TypeDefinitions* définies par le *Modèle d'information*.

Les QualifiedNames sont des dénominations non localisées qualifiées par un Namespace. Ils sont utilisés pour les *BrowseNames* des *Nœuds* et permettent aux mêmes noms d'être utilisés sans conflit par différents modèles d'information. Les *TypeDefinitions* ne sont pas autorisées à avoir des enfants ayant des *BrowseNames* dupliqués; cependant, les instances ne sont pas soumises à cette restriction.

5 Concept

5.1 Architecture du système

L'architecture du système représentée à la Figure 3 a déjà été définie dans l'IEC 62453-42:2016. Elle doit suivre l'architecture logicielle générale des outils FDT telle que définie dans l'IEC 62453-1 et peut être appliqué à tous les profils de mise en œuvre d'objets basés sur l'IEC 62453-2.

Le serveur OPC UA est une application-cadre. L'application-cadre interagit avec les DTM pour gérer les informations spécifiques au dispositif et pour interagir avec les dispositifs. Les applications-cadres peuvent prendre en charge différents cas d'utilisation, par exemple: service spécifique au fournisseur, contrôle des processus ou gestion des actifs. Les applications-cadres peuvent démarrer et terminer les DTM à la demande (par exemple uniquement si une valeur est lue à partir du dispositif concerné).

Le serveur OPC UA est équipé d'une interface serveur OPC UA. Dans le modèle d'information du serveur OPC UA, les informations de projet spécifiques à l'application peuvent être représentées (selon l'application-cadre) et les informations spécifiques au dispositif doivent être représentées. Le client OPC UA peut parcourir le modèle d'information et utiliser les services de l'interface OPC UA pour accéder aux informations représentées. Si des informations spécifiques au dispositif sont consultées, elles sont récupérées à partir des DTM.

Dans un scénario multiclient, l'application-cadre est responsable de la gestion de l'accès aux DTM. L'approche pour gérer l'accès de plusieurs clients consiste à utiliser "l'accès multiutilisateur" spécifié dans l'IEC 62453-42.

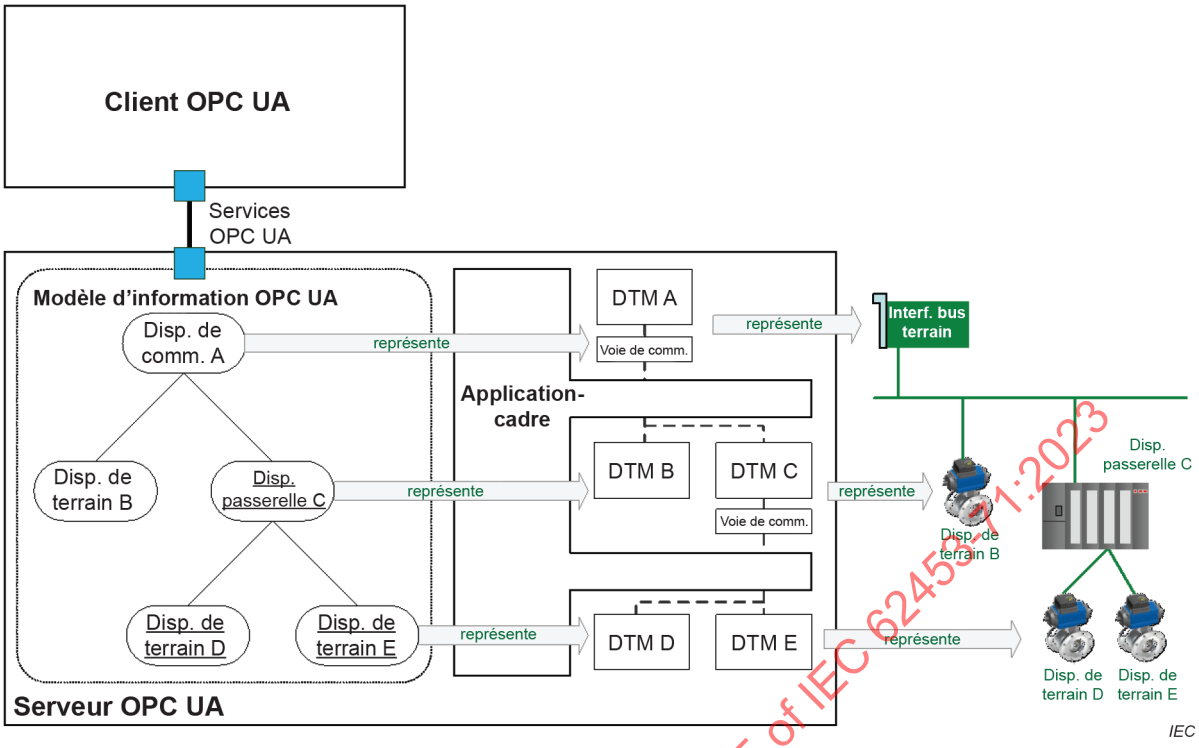


Figure 3 – Architecture du système conformément à l'IEC 62453-42

Comme le contenu du projet FDT et la gestion de l'accès multiclient sont spécifiques à l'application-cadre, le présent document est axé sur le mapping des informations du DTM au "Modèle de dispositif" de l'OPC UA DI. Une approche possible pour la présentation des informations du projet serait le "Modèle d'hôte d'intégration de dispositifs" conformément à la DI OPC UA.

6 ObjectTypes OPC UA spécifiques à la FDT

6.1 Généralités

Le modèle d'information de l'OPC UA pour outils FDT est basé sur l'IEC 62541-100. Le présent document définit un mapping pour les informations, les services et les types de données liés aux dispositifs.

Les services définis dans l'IEC 62541-100, mais pour lesquels aucun mapping n'est défini dans le présent document (par exemple le service de verrouillage), doivent être mis en œuvre par le serveur OPC UA conformément à la spécification de l'IEC 62541-100.

6.2 FdtDeviceType

Cet *ObjectType* OPC UA est la base d'un type de dispositif dérivé d'une définition de type de dispositif FDT. Tout type de dispositif spécifique au fabricant est dérivé de *FdtDeviceType*.

La Figure 4 représente une vue d'ensemble du *FdtDeviceType* avec ses Propriétés et ses ObjectTypes connexes. Il est défini de manière formelle dans le Tableau 12.

NOTE Specific DeviceType est un exemple illustrant la façon dont les types de dispositifs spécifiques à des dispositifs héritent du *FdtDeviceType*.

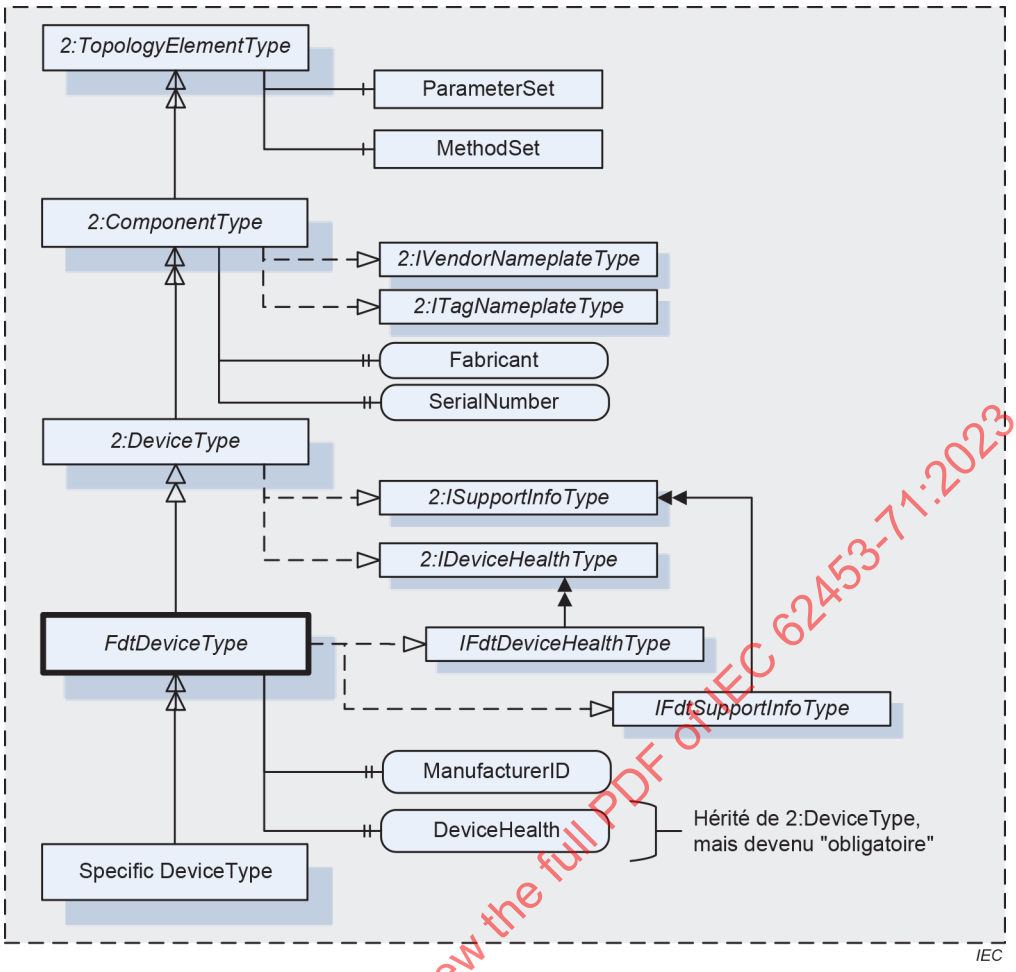


Figure 4 – Vue d'ensemble du FdtDeviceType

Le *FdtDeviceType* est défini de manière formelle dans le Tableau 12.

Tableau 12 – Définition de FdtDeviceType

Attribut	**Valeur**				
BrowseName	FdtDeviceType				
IsAbstract	True				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du 2:DeviceType défini dans l'IEC 62541-100					
HasInterface	ObjectType	IFdtDeviceHealthType		Défini en 6.4	
HasInterface	ObjectType	IFdtSupportInfoType		Défini en 6.5	
0:HasProperty	Variable	ManufacturerId	0:String	0:PropertyType	O
0:HasProperty	Variable	DeviceTypeId	0:String	0:PropertyType	O
0:HasProperty	Variable	DeviceTag	0:String	0:PropertyType	M
Repris du 2:IFdtDeviceHealthType					
0:HasComponent	Variable	2:DeviceHealth	2:DeviceHealth Enumeration	0:BaseDataVariable Type	M, RO [a]
[a] Cette définition de *DeviceHealth* remplace la définition de [1] [1].					

1 Les chiffres entre crochets renvoient à la Bibliographie.

Le *FdtDeviceType* est un type abstrait et ne peut pas être utilisé directement.

Le mapping des informations FDT avec FdtDeviceType est défini en 11.2.1.

défini en

6.3 FdtFunctionalGroupType

Le *FdtFunctionalGroupType* est utilisé pour représenter le groupage fonctionnel des méthodes et des paramètres. Il est défini de manière formelle dans le Tableau 13.

Tableau 13 – Définition de FdtFunctionalGroupType

Attribut	**Valeur**				
BrowseName	FdtFunctionalGroupType				
IsAbstract	False				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du 2:FunctionalGroupType défini dans l'IEC 62541-100:2015, 5.3					
0:HasProperty	Variable	ApplicationId	ApplicationIdEnumeration	0:PropertyType	O
0:HasProperty	Variable	SemanticInfo	SemanticInfoType	0:PropertyType	O

Le *FdtFunctionalGroupType* est un type concret et peut être utilisé directement.

6.4 Interface IFdtDeviceHealthType

Le *IFdtDeviceHealthType* définit des exigences supplémentaires pour l'interface *IDeviceHealthType*, qui est définie de manière formelle dans le Tableau 14.

Tableau 14 – Définition de IFdtDeviceHealthType

Attribut	**Valeur**				
BrowseName	IFdtDeviceHealthType				
IsAbstract	True				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type de l'interface 2:IDeviceHealthType [a].					
0:HasComponent	Variable	2:DeviceHealth	2:DeviceHealthEnumeration	0:BaseDataVariableType	M [b]
[a] Le IDeviceHealthType est expliqué en [1], 5.5.4.					
[b] Cette définition de *DeviceHealth* remplace la définition de [1].					

6.5 Interface IFdtSupportInfoType

6.5.1 Présentation

Le *IFdtSupportInfoType* définit des exigences supplémentaires pour l'interface *ISupportInfoType*, qui est définie de manière formelle dans le Tableau 15.

Tableau 15 – Définition de IFdtSupportInfoType

Attribut	Valeur				
BrowseName	IFdtSupportInfoType				
IsAbstract	True				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du 2:ISupportInfoType [a].					
[a] Le ISupportInfoType est expliqué en [1], 5.5.5.					

Les composantes du *IFdtSupportInfoType* ont des références supplémentaires définies dans le Tableau 16.

Tableau 16 – Sous-composantes supplémentaires du IFdtSupportInfoType

Chemin source	Références	Node Class	BrowseName	Type de données	TypeDefinition	Autres
2:Documentation	HasComponent	Objet	<FdtDocumentIdentifier>		FdtDocumentType	MP
2:ProtocolSupport	HasComponent	Objet	<FdtProtocolSupportIdentifier>		FdtDocumentType	MP

Les documents fournis pour un dispositif sont présentés comme des *Objets* organisés dans le dossier *Documentation*. Dans la plupart des cas, ils représentent un manuel de produit, qui peut exister sous la forme d'un ensemble de documents individuels. Le *BrowseName* de chaque *Objet* dans le Dossier *Documentation* se compose de l'étiquette du document qui peut être utilisée pour identifier le document.

Les fichiers de support de protocole (par exemple les fichiers GSD, les fichiers EDS) fournis pour un dispositif sont présentés comme des *Objets* organisés dans le dossier *ProtocolSupport*.

6.6 Types de documents

6.6.1 FdtDocumentType

Ce type abstrait décrit les informations associées à un document fourni par un DTM. Le *FdtDocumentType* est défini de manière formelle dans le Tableau 17.

Tableau 17 – Définition de FdtDocumentType

Attribut	Valeur				
BrowseName	FdtDocumentType				
IsAbstract	True				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du 0:BaseObjectType					
0:HasProperty	Variable	Classification	DocumentClassification	0:PropertyType	M
0:HasProperty	Variable	Help	0:String	0:PropertyType	O
0:HasProperty	Variable	Langue	0:String	0:PropertyType	O
0:HasProperty	Variable	MediaType	0:String	0:PropertyType	M
0:HasProperty	Variable	SemanticInfo	SemanticInfoType	0:PropertyType	O

Le *FdtDocumentType* est un type abstrait et ne peut pas être utilisé directement.

Le type de documents MIME est identifié par la propriété *MediaType* de chaque *Variable*.

6.6.2 FdtDocumentFile

Il s'agit d'un type permettant de représenter des documents qui sont fournis sous forme de fichiers. Il est défini de manière formelle dans le Tableau 18.

Tableau 18 – Définition de FdtDocumentFile

Attribut	**Valeur**				
BrowseName	FdtDocumentFile				
IsAbstract	False				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du FdtDocumentType					
0:HasComponent	Objet	Fichier		0:FileType	M

Le *FdtDocumentFile* est un type concret et peut être utilisé directement.

Le document peut être récupéré sur le serveur en utilisant les méthodes liées à l'objet de type *FileType*.

6.6.3 FdtDocumentUrl

Il s'agit d'un type permettant de représenter des documents qui sont fournis sous forme d'URL. Il est défini de manière formelle dans le Tableau 19.

Tableau 19 – Définition de FdtDocumentUrl

Attribut	**Valeur**				
BrowseName	FdtDocumentUrl				
IsAbstract	False				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du FdtDocumentType					
0:HasProperty	Variable	URL	0:String	0:PropertyType	M

Le *FdtDocumentUrl* est un type concret et peut être utilisé directement.

Le document réel peut être récupéré par le *Client* à l'URL.

6.7 FdtProtocolType

Ce type est utilisé pour spécifier les protocoles exigés ou pris en charge par un *FdtDeviceType* (voir Figure 5).

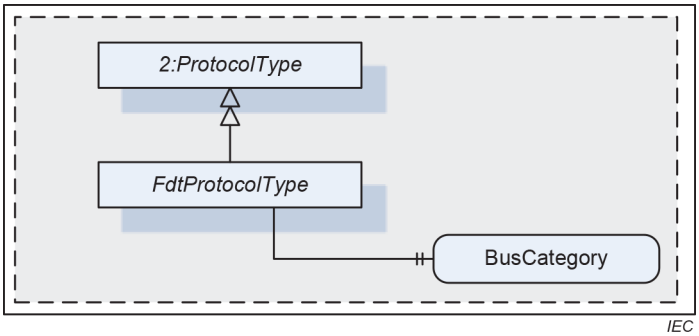


Figure 5 – Vue d'ensemble du FdtProtocolType

Le *FdtProtocolType* est défini de manière formelle dans le Tableau 20.

Tableau 20 – Définition de FdtProtocolType

Attribut	Valeur				
BrowseName	FdtProtocolType				
IsAbstract	True				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du 2:ProtocolType défini dans l'IEC 62541-100					
0:HasProperty	Variable	BusCategory	0:String	0:PropertyType	M

Le *FdtProtocolType* est un type abstrait et ne peut pas être utilisé directement.

La propriété *BusCategory* doit être utilisée par toute instance du *FdtProtocolType* ou de ses sous-types.

6.8 FdtTransferServiceType

Ce type est utilisé pour spécifier la prise en charge des services de transfert (voir Figure 6).

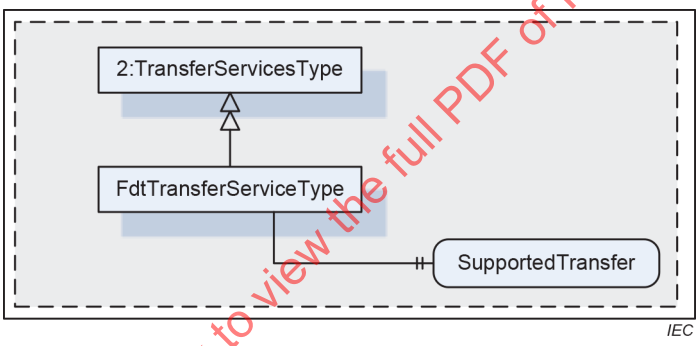


Figure 6 – Vue d'ensemble du FdtTransferServiceType

Le *FdtTransferServiceType* est défini de manière formelle dans le Tableau 21.

Tableau 21 – Définition de FdtTransferServiceType

Attribut	Valeur				
BrowseName	FdtTransferServiceType				
IsAbstract	False				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du 2:TransferServicesType					
0:HasProperty	Variable	SupportedTransfer	SupportedTransfer	0:PropertyType	M

Le *FdtTransferServiceType* est un type concret et peut être utilisé directement.

La propriété *SupportedTransfer* (définie en 9.4.11) est utilisée pour indiquer les services pris en charge par le dispositif concerné.

6.9 FdtIoSignalInfoType

Ce type est utilisé pour fournir des informations sur les signaux d'E/S disponibles sur le dispositif (voir Figure 7).

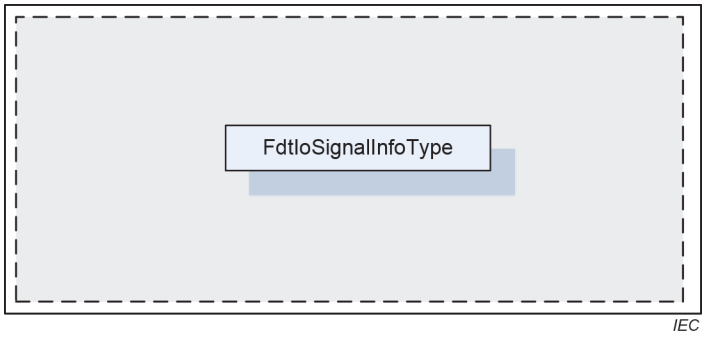


Figure 7 – Vue d'ensemble du FdtIoSignalInfoType

Le *FdtIoSignalInfoType* est défini de manière formelle dans le Tableau 22.

Tableau 22 – Définition de FdtIoSignalInfoType

Attribut	**Valeur**				
BrowseName	FdtIoSignalInfoType				
IsAbstract	False				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du BaseObjectType					
0:HasProperty	Variable	Description	0:String	0:PropertyType	O
0:HasProperty	Variable	FrameApplicationTag	0:String	0:PropertyType	M
0:HasProperty	Variable	IECDatatype	IECDatatype	0:PropertyType	M
0:HasProperty	Variable	IsLocked	0:Boolean	0:PropertyType	M
0:HasProperty	Variable	IsSafety	0:Boolean	0:PropertyType	M
0:HasProperty	Variable	RoutedIoSignalId	0:String	0:PropertyType	O
0:HasProperty	Variable	SemanticInfo	SemanticInfoType	0: PropertyType	O
0:HasProperty	Variable	SignalType	SignalTypeEnumeration	0:PropertyType	M
0:HasProperty	Variable	HasAlarmInfo	DataRefType	0:PropertyType	O
0:HasProperty	Variable	HasDeviceData	DataRefType	0:PropertyType	O
0:HasProperty	Variable	HasRange	DataRefType	0:PropertyType	O
0:HasProperty	Variable	HasSubstituteValue	DataRefType	0:PropertyType	O
0:HasProperty	Variable	HasUnit	DataRefType	0:PropertyType	O

Le *FdtIoSignalInfoType* est un type concret et peut être utilisé directement.

7 EventTypes OPC UA

7.1 Présentation

Afin de prendre en charge la piste de vérification, de nouveaux types d'événements de piste de vérification sont définis (voir Figure 8).

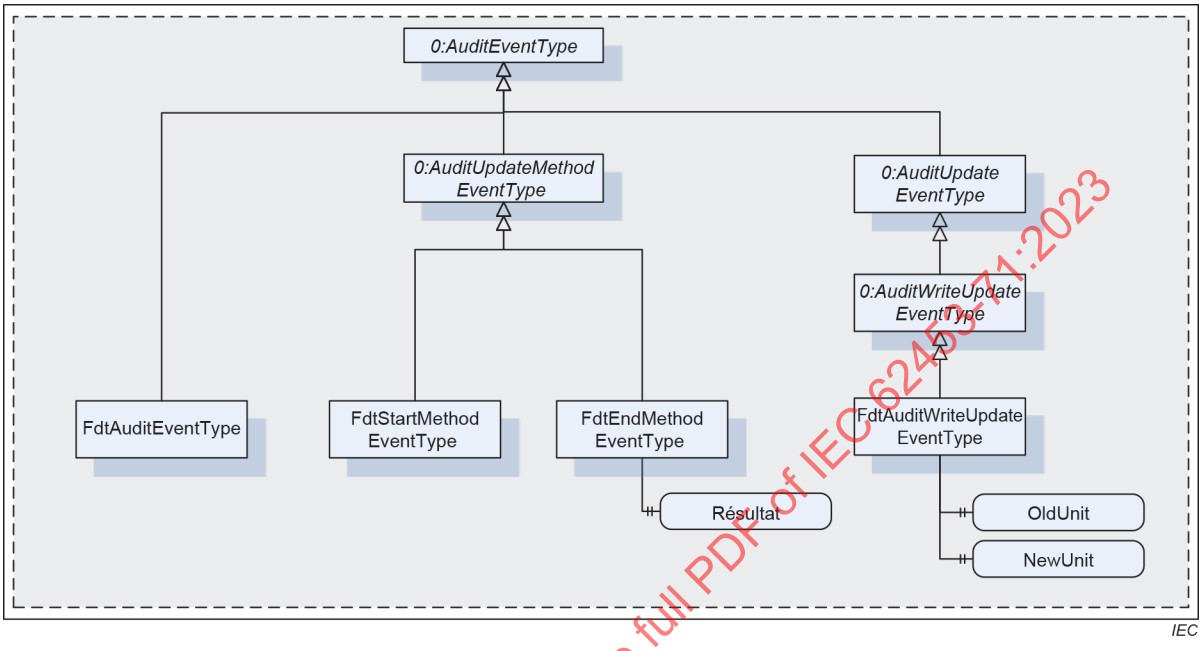


Figure 8 – Vue d'ensemble du type Évènement d'audit

7.2 FdtAuditEventType

Le *FdtAuditEventType* est un type d'événement pour événements généraux (par exemple pour la mise à jour de l'état du dispositif). Il est défini de manière formelle dans le Tableau 23.

Tableau 23 – Définition de FdtAuditEventType

Attribut	Valeur				
BrowseName	FdtAuditEventType				
IsAbstract	False				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du 0:AuditEventType défini dans l'IEC 62541-3, Modèle d'espace d'adressage, 9.5, ce qui signifie qu'il hérite des InstanceDeclarations de ce Nœud.					

7.3 FdtStartMethodEventType

Le *FdtStartMethodEventType* est un type d'événement destiné à indiquer le début de l'exécution d'une fonction de commande. Il est défini de manière formelle dans le Tableau 24.

Tableau 24 – Définition de FdtStartMethodEventType

Attribut		Valeur			
BrowseName		FdtStartMethodEventType			
IsAbstract		False			
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du 0:AuditUpdateMethodEventType défini en 7.2, ce qui signifie qu'il hérite des InstanceDeclarations de ce Nœud.					

7.4 FdtEndMethodEventType

Le *FdtEndMethodEventType* est un type d'événement destiné à indiquer la fin de l'exécution d'une fonction de commande. Il est défini de manière formelle dans le Tableau 25.

Tableau 25 – Définition de FdtEndMethodEventType

Attribut		Valeur			
BrowseName		FdtEndMethodEventType			
IsAbstract		False			
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du 0:AuditUpdateMethodEventType défini en 7.2, ce qui signifie qu'il hérite des InstanceDeclarations de ce Nœud.					
0:HasProperty	Variable	Résultat	FunctionExecutionResultState	0:PropertyType	M

La propriété Résultat est utilisée pour indiquer le résultat de l'exécution de la fonction.

7.5 FdtAuditWriteUpdateEventType

Le *FdtAuditWriteUpdateEventType* est utilisé pour indiquer la modification de la valeur d'un paramètre. Il est défini de manière formelle dans le Tableau 26.

Tableau 26 – Définition de FdtAuditWriteUpdateEventType

Attribut		Valeur			
BrowseName		FdtAuditWriteUpdateEventType			
IsAbstract		False			
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du 0:AuditWriteUpdateEventType défini dans l'IEC 62541-3, Modèle d'espace d'adressage, 9.27, ce qui signifie qu'il hérite des InstanceDeclarations de ce Nœud.					
0:HasProperty	Variable	OldUnit	0:String	0:PropertyType	O
0:HasProperty	Variable	NewUnit	0:String	0:PropertyType	O

La propriété *OldUnit* peut être utilisée pour indiquer l'unité de l'ancienne valeur. La propriété *NewUnit* peut être utilisée pour indiquer l'unité de la nouvelle valeur. Si le nœud source de l'événement a une unité technique, il est obligatoire de fournir ces propriétés.

8 VariableTypes OPC UA

8.1 FdtParameter

Un type spécifique *FdtParameter* est défini, il constitue une extension du *DataItemType* (voir Tableau 27).

Tableau 27 – Définition de FdtParameter

Attribut		Valeur			
BrowseName		FdtParameter			
IsAbstract		False			
ValueRank		−2 (−2 = n'importe laquelle)			
DataType		0:BaseDataType			
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du FdtParameter DataItemType/HasTypeDefinition					
0:HasProperty	Variable	DisplayFormat	0:String	0:PropertyType	O
0:HasProperty	Variable	AlarmType	AlarmType	0:PropertyType	O
0:HasProperty	Variable	RangeType	RangeType	0:PropertyType	O
0:HasProperty	Variable	SubstitutionType	SubstitutionType	0:PropertyType	O
0:HasProperty	Variable	ApplicationId	ApplicationIdEnumeration	0:PropertyType	O
0:HasProperty	Variable	0:EURange	0:Range	0:PropertyType	O
0:HasProperty	Variable	0:EngineeringUnits	0:EUInformation	0:PropertyType	O
0:HasProperty	Variable	0:EnumStrings	0:LocalizedText[]	0:PropertyType	O
0:HasProperty	Variable	SemanticInfo	SemanticInfoType	0:PropertyType	O
0:HasProperty	Variable	DataRef	DataRefType	0:PropertyType	O
0:HasProperty	Variable	AlarmDataRef	DataRefType	0:PropertyType	O
0:HasProperty	Variable	RangeDataRef	DataRefType	0:PropertyType	O
0:HasProperty	Variable	SubstituteDataRef	DataRefType	0:PropertyType	O
NonHierarchicalReferences					
HasIOSignalRef	ObjectType	FdtIoSignalInfoType			

Le *FdtParameter* est un type concret et peut être utilisé directement.

DisplayFormat est une chaîne de format pour les données numériques, conforme aux définitions du format de chaîne .NET.

Si le *DataType* du FdtParameter est un *DataType Énumération*, les *EnumStrings* sont utilisés conformément à l'IEC 62541-3.

EURange et *EngineeringUnits* sont utilisés comme défini dans l'IEC 62541-8 pour le type de variable *AnalogueItem*.

SemanticInfo fournit des informations sémantiques supplémentaires pour le paramètre (par exemple une référence à une définition dans un profil de dispositif).

Si le *FdtParameter* représente une information de données structurée, l'*ApplicationId* peut être utilisée pour catégoriser le cas d'utilisation de la structure de données.

Les propriétés *DataRefType* sont utilisées pour référencer des *FdtParameters* connexes:

– DataRef, fournit des références à d'autres données; les informations sur le type de référence sont fournies par la SemanticInfo du *DataRefType*;
– AlarmDataRef, fournit des références aux informations d'alarme;
– RangeDataRef, fournit des références aux informations de plage; et
– SubstituteDataRef, fournit des références pour remplacer les valeurs des paramètres.

Si le *FdtParameter* représente des informations d'alarme, *AlarmType* catégorise le type d'alarme disponible.

Si le *FdtParameter* représente des informations de plage, *RangeType* permet de distinguer si les données représentent une plage supérieure ou une plage inférieure.

Si le *FdtParameter* représente des informations de substitution, *SubstitutionType* spécifie la valeur qui doit être utilisée lorsqu'une valeur de substitution est nécessaire.

La référence *HasIOSignalRef* fournit une référence à la description d'un signal E/S qui correspond au *FdtParameter*.

9 DataTypes OPC UA

9.1 DataRefType

Le *DataRefType* fournit une référence à un élément de données. Il est défini de manière formelle dans le Tableau 28.

Tableau 28 – Structure du DataRefType

Name	Type	Description
DataRefType	structure	Référence à un élément de données.
DataId	0:String	Id de l'élément de données de référence.
SemanticInfo	SemanticInfoType	Signification de l'élément de données dans le contexte courant.

La représentation du *DataRefType* dans l'*AddressSpace* est définie dans le Tableau 29.

Tableau 29 – Définition de DataRefType

Attribut	Valeur				
BrowseName	DataRefType				
IsAbstract	False				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type de la 0:Structure définie dans l'IEC 62541-3.					

9.2 FdtDeviceClassificationType

Le *FdtDeviceClassificationType* est utilisé pour classer des dispositifs. Il est défini de manière formelle dans le Tableau 30.

Tableau 30 – Structure du FdtDeviceClassificationType

Name	Type	Description
FdtDeviceClassificationType	structure	Classification d'un dispositif conformément à l'IEC 62390, Annexe G.
ClassificationDomain	ClassificationDomainId	Groupes de domaines de classification des dispositifs.
DeviceClassification	ClassificationId	Identificateurs uniques selon la fonction principale du dispositif.

La représentation du *FdtDeviceClassificationType* dans l'*AddressSpace* est définie dans le Tableau 31.

Tableau 31 – Définition de FdtDeviceClassificationType

Attribut	Valeur				
BrowseName	FdtDeviceClassificationType				
IsAbstract	False				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type de la 0:Structure définie dans l'IEC 62541-3.					

9.3 SemanticInfoType

Ce type décrit les informations sémantiques associées aux données fournies par un DTM. Le *SemanticInfoType* est défini de manière formelle dans le Tableau 32.

Tableau 32 – Structure du SemanticInfoType

Name	Type	Description
SemanticInfoType	structure	Informations sémantiques associées aux données fournies par un DTM.
ApplicationDomain	0:String	Domaine d'application.
SemanticId	0:String	Identificateurs sémantiques dans le domaine.

La représentation du *SemanticInfoType* dans l'*AddressSpace* est définie dans le Tableau 33.

Tableau 33 – Définition de SemanticInfoType

Attribut	Valeur				
BrowseName	SemanticInfoType				
IsAbstract	False				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type de la 0:Structure définie dans l'IEC 62541-3.					

9.4 Datatypes Énumération

9.4.1 AlarmType

L'*AlarmType* est une énumération qui définit les types d'alarmes disponibles. Les valeurs doivent être mappées comme défini dans le Tableau 34.

Tableau 34 – Éléments d'AlarmType

Name	Valeur	Description
HighHighAlarm	0	Alarme de dépassement d'une valeur d'un processus de la limite "highhigh".
HighAlarm	1	Alarme de dépassement d'une valeur d'un processus de la limite "high".
LowLowAlarm	2	Alarme de chute d'une valeur d'un processus sous la limite "lowlow".
LowAlarm	3	Alarme de chute d'une valeur d'un processus sous la limite "low".

Sa représentation dans l'*AddressSpace* est définie dans le Tableau 35.

Tableau 35 – Définition d'AlarmType

Attribut	Valeur				
BrowseName	AlarmType				
IsAbstract	False				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du type Énumération défini dans l'IEC 62541-5					
0:HasProperty	Variable	0:EnumValues	0: EnumValueType[]	0:PropertyType	

9.4.2 ApplicationIdEnumeration

L'*ApplicationIdEnumeration* est une énumération qui définit l'utilisation du *FunctionalGroup*. Ses valeurs sont définies dans le Tableau 36.

Tableau 36 – Éléments d'ApplicationIdEnumeration

Name	Valeur	Description
AdjustSetValue	0	Ce groupe fonctionnel est utilisé pour régler la valeur de consigne.
AssetManagement	1	Ce groupe fonctionnel est utilisé pour la gestion des actifs.
AuditTrail	2	Ce groupe fonctionnel est utilisé pour la piste de vérification.
Configuration	3	Ce groupe fonctionnel est utilisé pour la configuration.
Diagnosis	4	Ce groupe fonctionnel est utilisé pour le diagnostic.
Force	5	Ce groupe fonctionnel est utilisé pour forcer des valeurs.
Observe	6	Ce groupe fonctionnel est utilisé pour observer un dispositif.
OfflineCompare	7	Ce groupe fonctionnel est utilisé pour comparer des données hors ligne de différents dispositifs.
OfflineParameterize	8	Ce groupe fonctionnel est utilisé pour le paramétrage hors ligne.
OnlineCompare	9	Ce groupe fonctionnel est utilisé pour comparer l'ensemble de données du dispositif à l'ensemble de données de l'instance.
OnlineParameterize	10	Ce groupe fonctionnel est utilisé pour le paramétrage en ligne.
Identify	11	Ce groupe fonctionnel est utilisé pour l'identification.
Calibration	12	Ce groupe fonctionnel est utilisé pour l'étalonnage.
MainOperation	13	Ce groupe fonctionnel est l'agrégation de tous les groupes fonctionnels.
NetworkManagement	14	Ce groupe fonctionnel est utilisé pour la gestion de réseau.

Sa représentation dans l'*AddressSpace* est définie dans le Tableau 37.

Tableau 37 – Définition d'ApplicationIdEnumeration

Attribut	Valeur				
BrowseName	ApplicationIdEnumeration				
IsAbstract	False				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du type Énumération défini dans l'IEC 62541-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType[]	0:PropertyType	

9.4.3 ClassificationDomainId

Le *ClassificationDomainId* est une énumération qui définit les domaines disponibles pour les classifications des dispositifs. Ses valeurs sont définies dans le Tableau 38.

Tableau 38 – Éléments de ClassificationDomainId

Name	Valeur	Description
PowerDistribution	0	Le domaine de classification est PowerDistribution.
MotionControl	1	Le domaine de classification est MotionControl.
Measurement	2	Le domaine de classification est Measurement.
OperatorInterface	3	Le domaine de classification est OperatorInterface.
ModulesAndControllers	4	Le domaine de classification est ModulesAndControllers.
Communication	5	Le domaine de classification est Communication.

Sa représentation dans l'*AddressSpace* est définie dans le Tableau 39.

Tableau 39 – Définition de ClassificationDomainId

Attribut	Valeur				
BrowseName	ClassificationDomainId				
IsAbstract	False				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du type Énumération défini dans l'IEC 62541-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType []	0:PropertyType	

9.4.4 ClassificationId

Le *ClassificationId* est une énumération qui définit les classifications disponibles pour les dispositifs. Ses valeurs sont définies dans le Tableau 40.

Tableau 40 – Éléments de ClassificationId

Name	Valeur	Description
Flow	0	La classification est Flow.
Level	1	La classification est Level.
Pressure	2	La classification est Pressure.
Temperature	3	La classification est Temperature.
Valve	4	La classification est Valve.
Positioner	5	La classification est Positioner.
Actuator	6	La classification est Actuator.
Nc_rc	7	La classification est Numeric Control/Robotic Control.
Encoder	8	La classification est Encoder.
SpeedDrive	9	La classification est SpeedDrive.
HMI	10	La classification est HMI.
AnalogInput	11	La classification est AnalogInput.
AnalogOutput	12	La classification est AnalogOutput.
DigitalInput	13	La classification est DigitalInput.
DigitalOutput	14	La classification est DigitalOutput.
ElectrochemicalAnalyser	15	La classification est ElectrochemicalAnalyser.
DtmSpecific	16	La classification est DtmSpecific.
Universal	17	La classification est Universal.
Analyser	18	La classification est Analyser.
RemoteIO	19	La classification est RemoteIO.
AnalogCombinedIO	20	La classification est AnalogCombinedIO.
DigitalCombinedIO	21	La classification est DigitalCombinedIO.
Recorder	22	La classification est Recorder.

Name	Valeur	Description
Controller	23	La classification est Controller.
Angle	24	La classification est Angle.
LimitSwitch	25	La classification est LimitSwitch.
Converter	26	La classification est Converter.
Motor	27	La classification est Motor.
Switchboard	28	La classification est Switchboard.
CircuitBreaker	29	La classification est CircuitBreaker.
PowerMonitoring	30	La classification est PowerMonitoring.
DistributionPanel	31	La classification est DistributionPanel.
Contactor	32	La classification est Contactor.
ProtectionStarter	33	La classification est ProtectionStarter.
SoftStarter	34	La classification est SoftStarter.
Drive	35	La classification est Drive.
AxisControl	36	La classification est AxisControl.
MotorControlCenter	37	La classification est MotorControlCenter.
ControlValve	38	La classification est ControlValve.
Electrical	39	La classification est Electrical.
Density	40	La classification est Density.
Quality	41	La classification est Quality.
SpeedOrRotaryFrequency	42	La classification est SpeedOrRotaryFrequency.
Radiation	43	La classification est Radiation.
WeightMass	44	La classification est WeightMass.
DistanceOrPositionPresence	45	La classification est DistanceOrPositionPresence.
PushButton	46	La classification est PushButton.
Joystick	47	La classification est Joystick.
Keypad	48	La classification est Keypad.
PilotLight	49	La classification est PilotLight.
StackLight	50	La classification est StackLight.
Display	51	La classification est Display.
CombinedButtonsAndLights	52	La classification est CombinedButtonsAndLights.
OperatorStation	53	La classification est OperatorStation.
GeneralInput	54	La classification est GeneralInput.
GeneralOutput	55	La classification est GeneralOutput.
CombinedInputOutput	56	La classification est CombinedInputOutput.
Relay	57	La classification est Relay.
Timer	58	La classification est Timer.
Scanner	59	La classification est Scanner.
ProgrammableController	60	La classification est ProgrammableController.
CommunicationAdapter	61	La classification est CommunicationAdapter.
Gateway	62	La classification est Gateway.

Sa représentation dans l'*AddressSpace* est définie dans le Tableau 41.

Tableau 41 – Définition de ClassificationId

Attribut	Valeur				
BrowseName	ClassificationId				
IsAbstract	False				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du type Énumération défini dans l'IEC 62541-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType[]	0:PropertyType	

9.4.5 DocumentClassification

La *DocumentClassification* est une énumération qui définit les classes disponibles de documents. Ses valeurs sont définies dans le Tableau 42.

Tableau 42 – Éléments de DocumentClassification

Name	Valeur	Description
Help	0	Ce document contient des informations d'aide.
TechnicalDocumentation	1	Ce document contient des informations techniques.
OrderingInformation	2	Ce document contient des informations relatives à la commande.
Miscellaneous	3	Ce document contient des informations générales.
TenderSpecification	4	Le document contient les spécifications de l'offre.

Sa représentation dans l'*AddressSpace* est définie dans le Tableau 43.

Tableau 43 – Définition de DocumentClassification

Attribut	Valeur				
BrowseName	DocumentClassification				
IsAbstract	False				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du type Énumération défini dans l'IEC 62541-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType[]	0:PropertyType	

9.4.6 FunctionExecutionResultState

Le *FunctionExecutionResultState* est une énumération qui définit le type d'états de résultats pour l'exécution d'une fonction fournie pour un dispositif. Ses valeurs sont définies dans le Tableau 44.

Tableau 44 – Éléments de FunctionExecutionResultState

Name	Valeur	Description
Cancel	0	La fonction a été annulée.
Success	1	L'exécution de la fonction s'est terminée avec succès.
Fail	2	L'exécution de la fonction a échoué.

Sa représentation dans l'*AddressSpace* est définie dans le Tableau 45.

Tableau 45 – Définition de FunctionExecutionResultState

Attribut		Valeur			
BrowseName		FunctionExecutionResultState			
IsAbstract		False			
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du type Énumération défini dans l'IEC 62541-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType[]	0:PropertyType	

9.4.7 IECDatatype

La *IECDatatype* est une énumération qui définit les types IEC d'un IOSignal. Ses valeurs sont définies dans le Tableau 46.

Tableau 46 – Éléments d'IECDatatype

Name	Valeur	Description
BOOL	0	1 bit -> vrai ou faux
SINT	1	Entier signé court (1 octet)
INT	2	Entier signé (2 octets)
DINT	3	Entier signé double (4 octets)
LINT	4	Entier signé long (8 octets)
USINT	5	Entier non signé court (1 octets)
UINT	6	Entier non signé (2 octets)
UDINT	7	Entier non signé double (4 octets)
ULINT	8	Entier non signé long (8 octets)
REAL	9	Virgule flottante (4 octets)
LREAL	10	Virgule flottante longue (8 octets)
TIME	11	Time
DATE	12	Date
TimeOfDay	13	Heure
DateAndTime	14	Date et heure
STRING	15	Chaîne de caractères à octet unique de longueur variable
BYTE	16	8 bits
WORD	17	16 bits
DWORD	18	32 bits
LWORD	19	64 bits
WSTRING	20	Chaîne de caractères à octet double de longueur variable

Sa représentation dans l'*AddressSpace* est définie dans le Tableau 47.

Tableau 47 – Définition de IECDatatype

Attribut		Valeur			
BrowseName		IECDatatype			
IsAbstract		False			
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du type Énumération défini dans l'IEC 62541-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType[]	0:PropertyType	

9.4.8 RangeType

Le *RangeType* est une énumération qui définit les types disponibles de limites de plage. Ses valeurs sont définies dans le Tableau 48.

Tableau 48 – Éléments de RangeType

Name	Valeur	Description
LowerRange	0	Les données du dispositif représentent une plage inférieure.
UpperRange	1	Les données du dispositif représentent une plage supérieure.

Sa représentation dans l'*AddressSpace* est définie dans le Tableau 49.

Tableau 49 – Définition de RangeType

Attribut	Valeur				
BrowseName	RangeType				
IsAbstract	False				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du type Énumération défini dans l'IEC 62541-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType[]	0:PropertyType	

9.4.9 SignalTypeEnumeration

La *SignalTypeEnumeration* est une énumération qui définit les types d'un signal E/S. Ses valeurs sont définies dans le Tableau 50.

Tableau 50 – Éléments de SignalTypeEnumeration

Name	Valeur	Description
Entrée	0	Signal d'entrée
Sortie	1	Signal de sortie

Sa représentation dans l'*AddressSpace* est définie dans le Tableau 51.

Tableau 51 – Définition de SignalTypeEnumeration

Attribut	Valeur				
BrowseName	SignalTypeEnumeration				
IsAbstract	False				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du type Énumération défini dans l'IEC 62541-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType[]	0:PropertyType	

9.4.10 SubstitutionType

Le *SubstitutionType* est une énumération qui définit les types de données de substitution. Ses valeurs sont définies dans le Tableau 52.

Tableau 52 – Éléments de SubstitutionType

Name	Valeur	Description
LastValue	0	La dernière valeur connue doit être utilisée.
LastValidValue	1	La dernière valeur valide doit être utilisée.
UpperRange	2	La plage supérieure doit être utilisée.
LowerRange	3	La plage inférieure doit être utilisée.

Sa représentation dans l'*AddressSpace* est définie dans le Tableau 53.

Tableau 53 – Définition de SubstitutionType

Attribut	Valeur				
BrowseName	SubstitutionType				
IsAbstract	False				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du type Énumération défini dans l'IEC 62541-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType[]	0:PropertyType	

9.4.11 SupportedTransfer

Le *SupportedTransfer* est une énumération qui définit les types de transferts fournis pour un dispositif. Ses valeurs sont définies dans le Tableau 54.

Tableau 54 – Éléments de SupportedTransfer

Name	Valeur	Description
None	0	Le DTM ne prend en charge ni le chargement ni le téléchargement.
OnlyDownload	1	Le DTM prend en charge uniquement l'écriture de données sur le dispositif. La lecture de données sur le dispositif n'est pas prise en charge.
OnlyUpload	2	Le DTM prend en charge uniquement la lecture de données sur le dispositif. L'écriture de données sur le dispositif n'est pas prise en charge.
BothUploadAndDownload	3	Le DTM prend en charge la lecture de valeurs à partir du dispositif et l'écriture de valeurs sur le dispositif.

Sa représentation dans l'*AddressSpace* est définie dans le Tableau 55.

Tableau 55 – Définition de SupportedTransfer

Attribut	Valeur				
BrowseName	SupportedTransfer				
IsAbstract	False				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**Autres**
Sous-type du type Énumération défini dans l'IEC 62541-5					
0:HasProperty	Variable	0:EnumValues	0:EnumValueType[]	0:PropertyType	

10 ReferenceType OPC UA – HasIOSignalRef

Le ReferenceType *HasIOSignalRef* est un *ReferenceType* concret qui peut être utilisé directement. Il s'agit d'un sous-type du *ReferenceType NonHierarchicalReferences*.

Le *TargetNode* d'une *Référence* du *ReferenceType HasIOSignalRef* fournit la description du signal E/S pour le *SourceNode*.

Le *SourceNode* de *Références* de ce type doit être un *FdtParameter*.

Le *TargetNode* de ce *ReferenceType* doit être un *FdtIoSignalInfoType*.

Le *HasIOSignalRef* est défini de manière formelle dans le Tableau 56.

Tableau 56 – Définition de HasIOSignalRef

Attributs	**Valeur**		
BrowseName	HasIOSignalRef		
InverseName	IsParameterOfIOSignal		
Symmetric	False		
IsAbstract	False		
Références	**NodeClass**	**BrowseName**	**Commentaire**
Sous-type des NonHierarchicalReferences défini dans l'IEC 62541-5.			

11 Mapping de DataTypes

11.1 Types de données primitives – Énumération DeviceHealth

Le datatype *DeviceHealthEnumeration*[2] doit être mappé avec le datatype FDT2 *DeviceStatus*. Les valeurs doivent être mappées comme défini dans le Tableau 57.

Tableau 57 – Mapping pour DeviceHealthEnumeration

Valeur de DeviceHealthEnumeration [a)]	**Valeur de DeviceStatus.StatusFlag FDT**
NORMAL_0	OK
FAILURE_1	Invalid
CHECK_FUNCTION_2	FunctionCheck
OFF_SPEC_3	OutOfSpecification
MAINTENANCE_REQUIRED_4	MaintenanceRequired
[a)] DeviceHealthEnumeration est expliqué en [1].	

2 DeviceHealthEnumeration est expliqué en [1].

11.2 Mapping avec types OPC DI

11.2.1 Type de dispositif

11.2.1.1 Généralités

L'IEC 62541-100 et l'IEC 62453 utilisent des approches différentes en ce qui concerne la représentation des données spécifiques au type de dispositif. Certaines des données définies dans l'OPC UA pour la représentation du type de dispositif sont disponibles uniquement dans les données spécifiques à l'instance des DTM (voir Figure 9).

Les données FDT peuvent être mappées soit avec des attributs (en-tête de tableau "*Attribute*"), soit avec des propriétés et autres *Nœuds* enfants (en-tête de tableau "BrowseName") du *Nœud* OPC UA correspondant.

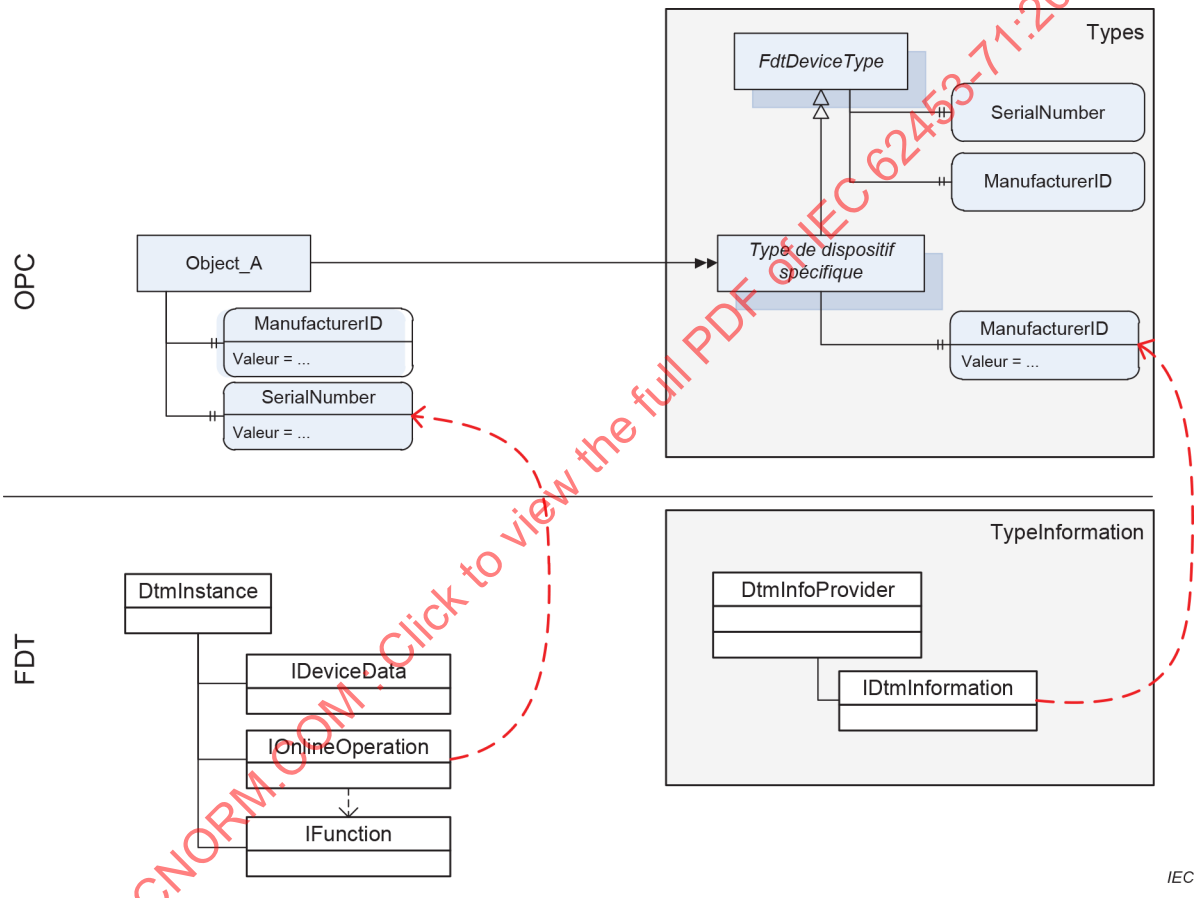


Figure 9 – Exemple de sources d'informations DeviceType

Pour cette raison, les données spécifiques au type de dispositif d'OPC sont mappées avec des données de plusieurs interfaces du DTM (voir Tableau 58).

11.2.1.2 Mapping pour DeviceType (Objets normalisés "Types")

Tous les types de dispositifs fournis par les DTM dans IDtmInformation doivent être représentés comme des sous-types du *FdtDeviceType* dans l'Objet normalisé "Types" (voir Tableau 58).

Tableau 58 – Mapping du DeviceType

OPC	FDT			
Attribut	**Interface**	**Méthode**	**Membre des données**	**Description**
BrowseName	IDtmInformation	<>GetSupported Types	DeviceTypeInfo.Name	
DisplayName	IDtmInformation	<>GetSupported Types	DeviceTypeInfo.Name	
BrowseName	**Interface**	**Méthode**	**Membre des données**	**Description**
2:SerialNumber				Mapping pour dispositif en ligne uniquement, voir Tableau 61
2:RevisionCounter				Non pris en charge. Valeur toujours configurée sur –1
2:Manufacturer	IDtmInformation	<>GetSupported Types	DeviceTypeInfo.Product ManufacturerName	
ManufacturerId	IDtmInformation	GetDeviceIdentInfo	DeviceIdentInfo. ManufacturerId	
2:Model	IDtmInformation	<>GetSupported Types	DeviceTypeInfo.ProductName	
2:DeviceManual				Tous les documents sont fournis dans le dossier Documentation
2:DeviceRevision	IDtmInformation	<>GetSupported Types	DeviceTypeInfo.Product Revision	
2:SoftwareRevision	IDtmInformation	GetDeviceIdentInfo	DeviceIdentInfo.Software Revision	
2:HardwareRevision	IDtmInformation	GetDeviceIdentInfo	DeviceIdentInfo. HardwareRevision	
2:DeviceClass(O)	IDtmInformation	<>GetSupported Types	DeviceTypeInfo.Device Classifications[0].DomainId + ":" + DeviceTypeInfo.Device Classifications[0].Id	La valeur est une concaténation des noms des membres de l'énumération pour DomainId et Id.
DeviceTypeId	IDtmInformation	<>GetSupported Types	DeviceTypeInfo.Id	
2:ManufacturerUri(O)				Non pris en charge.
2:ProductCode(O)				Non pris en charge.
2:ProductInstanceUri(O)				Non pris en charge.
<CP_Identifier>(O)	IDtmInformation	<>GetSupported Types	TypeInfo.BusCategories[].Category¬Type == exigé → <Identificateur> = TypeInfo.BusCategories[].Protocol¬Name	Référence HasComponent du type RequiredProtocol.
	IDtmInformation IChannels	<>GetSupported Types IChannels. Communication Channels	TypeInfo.BusCategories[].Category¬Type == pris en charge → <Identificateur> = CommunicationChannel Item.Id	Référence HasComponent du SupportedProtocol de type. Si une Voie exige plusieurs protocoles, plusieurs points de connexion sont prévus, un numéro supplémentaire est concaténé à l'identificateur de la voie.
0:Icon	IDtmInformation	GetFdtIcon()	Icon	Voir l'explication après le tableau
Repris du IFdtDeviceHealthType				
2:DeviceHealth				Mapping pour dispositif en ligne uniquement, voir Tableau 61
2:DeviceHealthAlarms(O)	–	–	–	Aucun mapping défini.

OPC	FDT			
BrowseName	**Interface**	**Méthode**	**Membre des données**	**Description**
Repris du IFdtSupportInfoType				
2:DeviceTypeImage (O)				Ce dossier contient les éléments définis en 11.2.8.1.
2:Documentation (O)	IDtmInformation	<>GetSupported Types	DeviceTypeInfo. Documents	Ce dossier contient des éléments qui représentent le membre de données FDT représenté, tel que défini en 6.3
2:ProtocolSupport (O)	INetworkData	GetNetworkDataInfo	NetworkData. DeviceInformation Documents	Ce dossier est fourni comme défini en 11.2.8.2.
2:ImageSet (O)				Non pris en charge.

Un DTM peut fournir plusieurs icônes (liste d'icônes). La règle de sélection de l'icône et d'extraction d'une image (d'une résolution spécifique) à partir de l'icône est spécifique à l'application-cadre. La sélection du format d'image en général est spécifique à la mise en œuvre, tous les types définis dans l'IEC 62541-3 sont autorisés.

11.2.1.3 Mapping pour Dispositif (Objet normalisé "Objects") hors ligne

Les informations relatives à un dispositif (instance) sont basées sur les informations fournies par un DTM (instance). Le mapping du nœud du dispositif est défini dans le Tableau 59.

Tableau 59 – Mapping des informations de dispositif

OPC	FDT			
Attribut	**Interface**	**Méthode**	**Membre des données**	**Description**
TypeReference	–	–	–	NodeId du DeviceType
BrowseName	IDtm	DtmSystemGuiLabel		
DisplayName	IDtm	DtmSystemGuiLabel		

Les paramètres du dispositif offrent des informations d'identification hors ligne basées sur les informations de type DTM destinées à l'identification. Ces données peuvent fournir des valeurs ou des expressions régulières pour définir des plages de valeurs prises en charge.

Dans l'appel à IDtmInformation.GetDeviceIdentInfo(), un protocole doit être spécifié. La première entrée de la liste de INetworkData.ActiveProtocols doit être utilisée.

GetDeviceIdentInfo() peut renvoyer une liste de DeviceIdentInfo où la première entrée est utilisée pour les paramètres définis ci-après. L'exposition des autres entrées de la liste est spécifique à la mise en œuvre.

Tableau 60 – Mapping des paramètres du dispositif hors ligne

OPC	FDT			
BrowseName	**Interface**	**Méthode**	**Membre des données**	**Description**
ManufacturerId	IDtmInformation	GetDeviceIdentInfo()	DeviceIdentInfo[0].Get ManufacturerId()	
DeviceTypeId		IDtmInformation	GetDeviceIdentInfo()	DeviceIdentInfo[0]. GetDeviceTypeId()
2:NetworkAddress	INetworkData	GetAddressInfo()	AddressInfo.Device Addresses[].Address	matrice de chaînes contenant toutes les adresses fournies
DeviceTag	-/-	-/-	(valeur par défaut)	
2:Serial Number	-/-	-/-	(valeur par défaut)	
2:SoftwareRevision	IDtmInformation	GetDeviceIdentInfo()	DeviceIdentInfo[0].Get SoftwareRevision()	
2:HardwareRevisio n	IDtmInformation	GetDeviceIdentInfo()	DeviceIdentInfo[0].Get HardwareRevision()	
Contenu du FunctionalGroup "Identification" (voir 11.2.5)				
ProtocolId	IDtmInformation	GetDeviceIdentInfo()	DeviceIdentInfo[0].Get ProtocolId()	
ProtocolSpecific Identification	IDtmInformation	GetDeviceIdentInfo()	DeviceIdentInfo[0]. GetProtocolSpecific Properties()	
DeviceSpecific-Identification	IDtmInformation	GetDeviceIdentInfo()	DeviceIdentInfo[0].Device SpecificProperties	

11.2.1.4 Mapping pour Dispositif (Objet normalisé "Objects" → référence en ligne) en ligne

Les paramètres du dispositif fournissent des informations d'identification basées sur les données lues en ligne à partir du dispositif. Ces données peuvent être différentes de celles fournies dans le *DeviceType*.

Tableau 61 – Mapping des paramètres du dispositif en ligne

OPC	FDT			
BrowseName	**Interface**	**Méthode**	**Membre des données**	**Description**
ManufacturerId	IHardwareInformation	<>HardwareScan()	DeviceScanInfo.Get ManufacturerId()	
DeviceTypeId	IHardwareInformation	<>HardwareScan()	DeviceScanInfo.GetDeviceT ypeId()	
2:Network Address	IHardwareInformation	<>HardwareScan()	DeviceScanInfo.Get Address()	
DeviceTag	IHardwareInformation	<>HardwareScan()	DeviceScanInfo.GetTag()	
2:DeviceHealth	IOnlineOperation	<>ReadDevice-Status()	DeviceStatus.StatusFlag	Pour le mapping des valeurs de l'énumération, voir 11.1.
2:Serial Number	IHardwareInformation	<>HardwareScan()	DeviceScanInfo.Serial Number()	
2:Software Revision	IHardwareInformation	<>HardwareScan()	DeviceScanInfo.Get SoftwareRevision()	
2:Hardware Revision	IHardwareInformation	<>HardwareScan()	DeviceScanInfo.Get HardwareRevision()	
Contenu du FunctionalGroup "Identification" (voir 11.2.5)				
ProtocolId	IHardwareInformation	<>HardwareScan()	DeviceScanInfo.Get ProtocolId()	
ProtocolSpecific Identification	IHardwareInformation	<>HardwareScan()	DeviceScanInfo.Get ProtocolSpecificProperties()	
DeviceSpecific-Identification	IHardwareInformation	<>HardwareScan()	DeviceScanInfo.Device SpecificProperties	

11.2.2 TopologyElementType

Les informations fournies par un DTM avec IFunction.FunctionInfo et IData.DataInfo sont mappées dans le *MethodSet* et dans le *ParameterSet*. Pour les deux types d'informations (Commandes et Paramètres), des références dans les nœuds *FunctionalGroup* sont également créées, afin qu'un Client OPC puisse représenter une interface utilisateur commune (voir Figure 10).

Le *BrowseName* et le *DisplayName* des Nœuds *FunctionalGroup* sont basés sur le groupage dans IFunction.FunctionInfo (FunctionGroup) et IData.DataInfo (DataGroup).

Les informations sur les signaux E/S sont fournies dans un dossier ProcessDataSet, qui contient des nœuds *FdtIoSignalInfoType*. Les nœuds *FdtIoSignalInfoType* sont référencés par les nœuds *FdtParameter* correspondants.

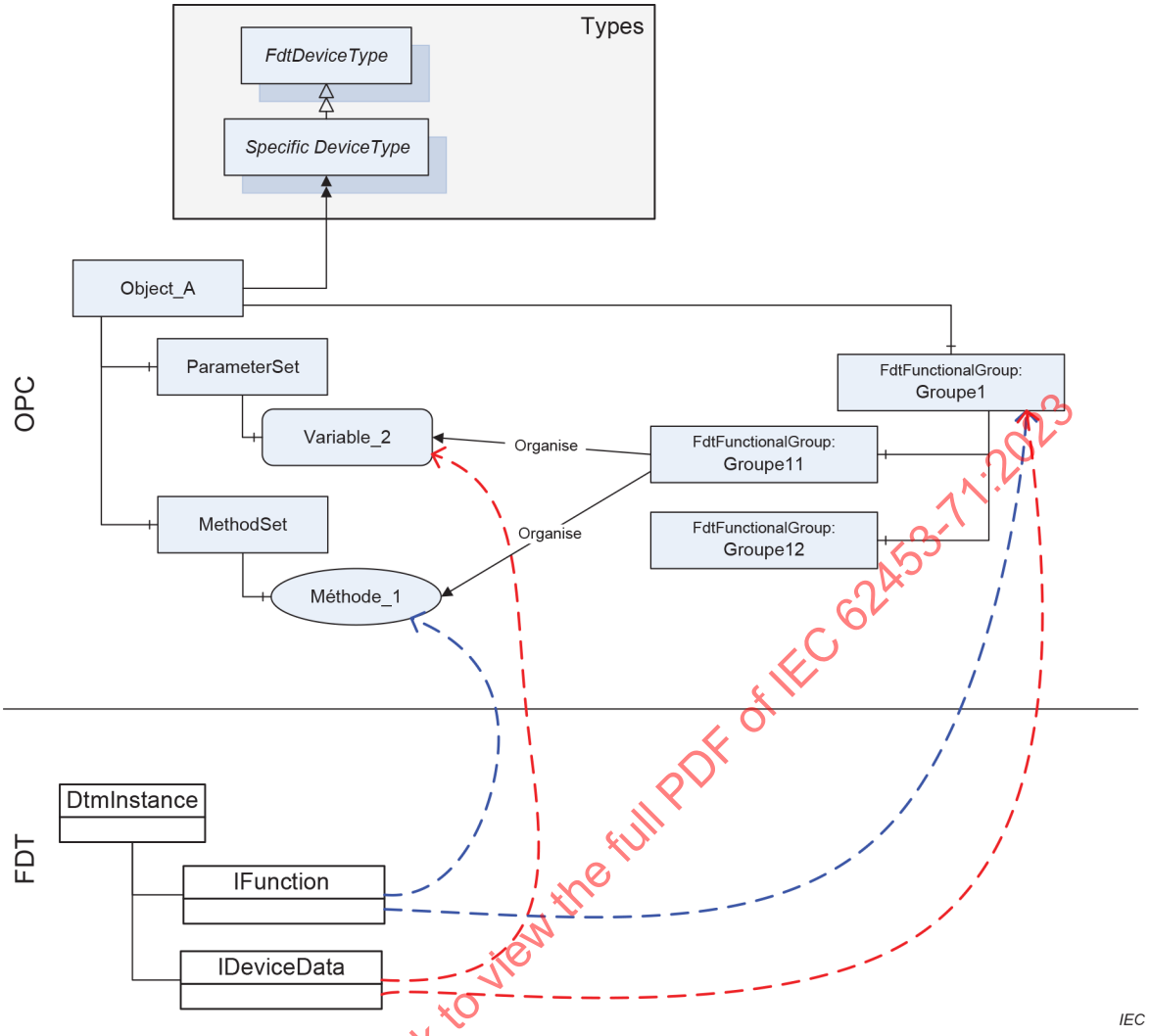


Figure 10 – Exemple de sources d'informations TopologyType