

INTERNATIONAL STANDARD

NORME INTERNATIONALE



**Field Device Integration (FDI) –
Part 5: FDI Information Model**

**Intégration des appareils de terrain (FDI) –
Partie 5: Modèle d'Information FDI**



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2015 IEC, Geneva, Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either IEC or IEC's member National Committee in the country of the requester. If you have any questions about IEC copyright or have an enquiry about obtaining additional rights to this publication, please contact the address below or your local IEC member National Committee for further information.

Droits de reproduction réservés. Sauf indication contraire, aucune partie de cette publication ne peut être reproduite ni utilisée sous quelque forme que ce soit et par aucun procédé, électronique ou mécanique, y compris la photocopie et les microfilms, sans l'accord écrit de l'IEC ou du Comité national de l'IEC du pays du demandeur. Si vous avez des questions sur le copyright de l'IEC ou si vous désirez obtenir des droits supplémentaires sur cette publication, utilisez les coordonnées ci-après ou contactez le Comité national de l'IEC de votre pays de résidence.

IEC Central Office
3, rue de Varembe
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00
info@iec.ch
www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

IEC Catalogue - webstore.iec.ch/catalogue

The stand-alone application for consulting the entire bibliographical information on IEC International Standards, Technical Specifications, Technical Reports and other documents. Available for PC, Mac OS, Android Tablets and iPad.

IEC publications search - www.iec.ch/searchpub

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 It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished

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

Electropedia - www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 30 000 terms and definitions in English and French, with equivalent terms in 15 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary

More than 60 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

IEC Customer Service Centre - webstore.iec.ch/csc

If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: csc@iec.ch.

A propos de l'IEC

La Commission Electrotechnique Internationale (IEC) est la première organisation mondiale qui élabore et publie des Normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications IEC

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

Catalogue IEC - webstore.iec.ch/catalogue

Application autonome pour consulter tous les renseignements bibliographiques sur les Normes internationales, Spécifications techniques, Rapports techniques et autres documents de l'IEC. Disponible pour PC, Mac OS, tablettes Android et iPad.

Recherche de publications IEC - www.iec.ch/searchpub

La recherche avancée permet de trouver des publications IEC en utilisant différents critères (numéro de référence, texte, comité d'études,...). Elle donne aussi des informations sur les projets et les publications remplacées ou retirées.

IEC Just Published - webstore.iec.ch/justpublished

Restez informé sur les nouvelles publications IEC. Just Published détaille les nouvelles publications parues. Disponible en ligne et aussi une fois par mois par email.

Electropedia - www.electropedia.org

Le premier dictionnaire en ligne de termes électroniques et électriques. Il contient plus de 30 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans 15 langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (IEV) en ligne.

Glossaire IEC - std.iec.ch/glossary

Plus de 60 000 entrées terminologiques électrotechniques, en anglais et en français, extraites des articles Termes et Définitions des publications IEC parues depuis 2002. Plus certaines entrées antérieures extraites des publications des CE 37, 77, 86 et CISPR de l'IEC.

Service Clients - webstore.iec.ch/csc

Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions contactez-nous: csc@iec.ch.

INTERNATIONAL STANDARD

NORME INTERNATIONALE



**Field Device Integration (FDI) –
Part 5: FDI Information Model**

**Intégration des appareils de terrain (FDI) –
Partie 5: Modèle d'Information FDI**

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

COMMISSION
ELECTROTECHNIQUE
INTERNATIONALE

ICS 25.040.40; 35.100

ISBN 978-2-8322-2636-0

**Warning! Make sure that you obtained this publication from an authorized distributor.
Attention! Veuillez vous assurer que vous avez obtenu cette publication via un distributeur agréé.**

CONTENTS

FOREWORD.....	6
INTRODUCTION.....	8
1 Scope.....	9
2 Normative references	10
3 Terms, definitions, abbreviated terms, acronyms and conventions	10
3.1 Terms and definitions.....	10
3.2 Abbreviated terms and acronyms	11
3.3 Conventions for graphical notation	11
4 Overview of OPC Unified Architecture	13
4.1 General.....	13
4.2 Overview of OPC UA Devices	13
5 Concepts	15
5.1 General.....	15
5.2 Device topology	15
5.3 Online/offline	17
5.4 Catalogue (Type Definitions).....	18
5.5 Communication	18
6 AddressSpace organization	18
7 Device Model for FDI	19
7.1 General.....	19
7.2 Online/offline	19
7.3 Device health.....	20
7.4 User interface elements	20
7.4.1 General	20
7.4.2 UI Description Type	21
7.4.3 UI Plug-in Type.....	21
7.5 Type-specific support information	23
7.6 Actions	23
7.6.1 Overview	23
7.6.2 Action Type	25
7.6.3 ActionService Type.....	25
7.6.4 ActionService Object	26
7.6.5 InvokeAction Method	26
7.6.6 RespondAction Method	27
7.6.7 AbortAction Method	28
8 Network and connectivity.....	28
9 Utility functions	29
9.1 Overview.....	29
9.2 Locking	29
9.3 EditContext.....	29
9.3.1 Overview	29
9.3.2 EditContext Type	30
9.3.3 EditContext Object.....	30
9.3.4 GetEditContext Method	30
9.3.5 RegisterNodes Method	31

9.3.6	Apply Method	32
9.3.7	Reset Method	33
9.3.8	Discard Method	34
9.4	Direct Device Access	34
9.4.1	General	34
9.4.2	DirectDeviceAccess Type	35
9.4.3	DirectDeviceAccess Object	36
9.4.4	InitDirectAccess Method	36
9.4.5	EndDirectAccess Method	37
9.4.6	Transfer Method	37
10	Parameter Types	38
10.1	General	38
10.2	ScalingFactor Property	39
10.3	Min_Max_Values Property	39
11	FDI StatusCodes	40
12	Specialized topology elements	40
13	Auditing	41
13.1	General	41
13.2	FDI Client-provided context information	41
13.3	LogAuditTrailMessage Method	41
14	FDI Server Version	42
15	Mapping FDI Package information to the FDI Information Model	42
15.1	General	42
15.2	Localization	43
15.2.1	Localized text	43
15.2.2	Engineering units	43
15.3	Device	43
15.3.1	General	43
15.3.2	Mapping to Attributes to a specific DeviceType Node	43
15.3.3	Mapping to Properties	43
15.3.4	Mapping to ParameterSet	44
15.3.5	Mapping to Functional Groups	44
15.3.6	Mapping to DeviceTypeImage	44
15.3.7	Mapping to Documentation	44
15.3.8	Mapping to ProtocolSupport	44
15.3.9	Mapping to ImageSet	44
15.3.10	Mapping to ActionSet	45
15.3.11	Mapping to MethodSet	45
15.4	Block	45
15.4.1	General	45
15.4.2	Mapping to Attributes	45
15.4.3	Mapping to ParameterSet	45
15.4.4	Mapping to Functional Groups	45
15.4.5	Mapping to ActionSet	46
15.4.6	Mapping to MethodSet	46
15.4.7	Instantiation rules	46
15.5	Parameter	46
15.5.1	General	46

15.5.2	Private Parameters	49
15.5.3	MIN_Value and MAX_Value	49
15.5.4	Engineering units	49
15.5.5	Enumerated Parameters	50
15.5.6	Bit-enumerated Parameters	50
15.5.7	Representation of records	50
15.5.8	Representation of arrays, and lists of Parameters with simple data types	51
15.5.9	Representation of values arrays, and lists of RECORD Parameters	52
15.5.10	Representation of COLLECTION and REFERENCE ARRAY	52
15.5.11	SCALING_FACTOR	52
15.6	Functional Groups	53
15.7	AXIS elements in UIDs	53
15.8	Actions	54
15.9	UIPs	54
15.10	Protocols, Networks and Connection Points	54
Annex A (normative)	Namespace and Mappings	55
Bibliography		56
Figure 1	– FDI architecture diagram	9
Figure 2	– OPC UA Graphical Notation for NodeClasses	11
Figure 3	– OPC UA Graphical Notation for References	11
Figure 4	– OPC UA Graphical Notation Example	12
Figure 5	– Optimized Type Reference	12
Figure 6	– OPC UA Devices Example: Functional Groups	14
Figure 7	– OPC UA Devices example: Configurable components	15
Figure 8	– Example of an automation system	16
Figure 9	– Example of a Device topology	17
Figure 10	– Example Device Types representing a catalogue	18
Figure 11	– Online component for access to device data	20
Figure 12	– Hierarchy of user interface Types	21
Figure 13	– Integration of Actions within a TopologyElement	24
Figure 14	– Action Service	26
Figure 15	– EditContext type and instance	30
Figure 16	– DirectDeviceAccessType	35
Figure 17	– DirectDeviceAccess instance	36
Figure 18	– OPC UA VariableTypes including OPC UA DataAccess	39
Figure 19	– Example: Complex variable representing a RECORD	51
Figure 20	– Complex variable representing a VALUE_ARRAY of RECORDs	52
Table 1	– UIDescriptionType Definition	21
Table 2	– UIPlugInType Definition	22
Table 3	– TopologyElementType with additions for Actions	24
Table 4	– FunctionalGroupType with additions for Actions	25
Table 5	– ActionType Definition	25
Table 6	– ActionServiceType Definition	25

Table 7 – InvokeAction Method Arguments	27
Table 8 – InvokeAction Method AddressSpace Definition	27
Table 9 – RespondAction Method Arguments	27
Table 10 – RespondAction Method AddressSpace Definition	28
Table 11 – AbortAction Method Arguments	28
Table 12 – AbortAction Method AddressSpace Definition	28
Table 13 – EditContextType Definition	30
Table 14 – GetEditContext Method Arguments	31
Table 15 – GetEditContext Method AddressSpace Definition	31
Table 16 – RegisterNodes Method Arguments	31
Table 17 – RegisterNodes Method AddressSpace Definition	32
Table 18 – RegistrationParameters DataType Structure	32
Table 19 – RegisterNodesResult DataType Structure	32
Table 20 – Apply Method Arguments	33
Table 21 – Apply Method AddressSpace Definition	33
Table 22 – ApplyResult DataType Structure	33
Table 23 – Reset Method Arguments	34
Table 24 – Reset Method AddressSpace Definition	34
Table 25 – Discard Method Arguments	34
Table 26 – Discard Method AddressSpace Definition	34
Table 27 – DirectDeviceAccessType Definition	35
Table 28 – DirectDeviceAccess Instance Definition	36
Table 29 – InitDirectAccess Method Arguments	37
Table 30 – InitDirectAccess Method AddressSpace Definition	37
Table 31 – EndDirectAccess Method Arguments	37
Table 32 – EndDirectAccess Method AddressSpace Definition	37
Table 33 – Transfer Method Arguments	38
Table 34 – Transfer Method AddressSpace Definition	38
Table 35 – ScalingFactor Property Definition	39
Table 36 – Min_Max_Values Property Definition	40
Table 37 – Variant_Range DataType Structure	40
Table 38 – Variant_Range Definition	40
Table 39 – Good operation level result codes	40
Table 40 – LogAuditTrailMessage Method Arguments	42
Table 41 – LogAuditTrailMessage Method AddressSpace Definition	42
Table 42 – FDIServerVersion Property Definition	42
Table 43 – DeviceType Property Mapping	44
Table 44 – Setting OPC UA Variable Attributes from EDDL variable attributes	47
Table 45 – Correspondence between EDDL and OPC UA standard data types	47

INTERNATIONAL ELECTROTECHNICAL COMMISSION

FIELD DEVICE INTEGRATION (FDI) –**Part 5: FDI Information Model****FOREWORD**

- 1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
- 2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
- 3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
- 4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
- 5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
- 6) All users should ensure that they have the latest edition of this publication.
- 7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
- 8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

International Standard IEC 62769-5 has been prepared by subcommittee 65E: Devices and integration in enterprise systems, of IEC technical committee 65: Industrial-process measurement, control and automation.

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

CDV	Report on voting
65E/348/CDV	65E/425/RVD

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

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

A list of all parts in the IEC 62769 series, published under the general title *Field Device Integration (FDI)*, can be found on the IEC website.

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

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

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

INTRODUCTION

The International Electrotechnical Commission (IEC) draws attention to the fact that it is claimed that compliance with this document may involve the use of patents concerning

- a) Method for the Supplying and Installation of Device-Specific Functionalities, see Patent Family DE10357276;
- b) Method and device for accessing a functional module of automation system, see Patent Family EP2182418;
- c) Methods and apparatus to reduce memory requirements for process control system software applications, see Patent Family US2013232186;
- d) Extensible Device Object Model, see Patent Family US12/893,680.

IEC takes no position concerning the evidence, validity and scope of this patent right.

The holders of these patent rights have assured the IEC that he/she is willing to negotiate licences either free of charge or under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with IEC. Information may be obtained from:

- a) ABB Research Ltd
Claes Ryttoft
Affolterstrasse 4
Zurich, 8050
Switzerland
- b) Phoenix Contact GmbH & Co KG
Intellectual Property, Licenses & Standards
Flachsmarktstrasse 8, 32825 Blomberg
Germany
- c) Fisher Controls International LLC
John Dilger, Emerson Process Management LLLP
301 S. 1st Avenue, Marshalltown, Iowa 50158
USA
- d) Rockwell Automation Technologies, Inc.
1 Allen-Bradley Drive
Mayfield Heights, Ohio 44124
USA

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. IEC shall not be held responsible for identifying any or all such patent rights.

ISO (www.iso.org/patents) and IEC (<http://patents.iec.ch>) maintain on-line data bases of patents relevant to their standards. Users are encouraged to consult the data bases for the most up to date information concerning patents.

FIELD DEVICE INTEGRATION (FDI) –

Part 5: FDI Information Model

1 Scope

This part of IEC 62769 defines the FDI Information Model. One of the main tasks of the Information Model is to reflect the topology of the automation system. Therefore it represents the devices of the automation system as well as the connecting communication networks including their properties, relationships, and the operations that can be performed on them. The types in the AddressSpace of the FDI Server constitute some kind of catalogue, which is built from FDI Packages.

The fundamental types for the FDI Information Model are well defined in OPC UA for Devices (IEC 62541-100). The FDI Information Model specifies extensions for a few special cases and otherwise explains how these types are used and how the contents are built from elements of DevicePackages.

The overall FDI architecture is illustrated in Figure 1. The architectural components that are within the scope of this document have been highlighted in this illustration.

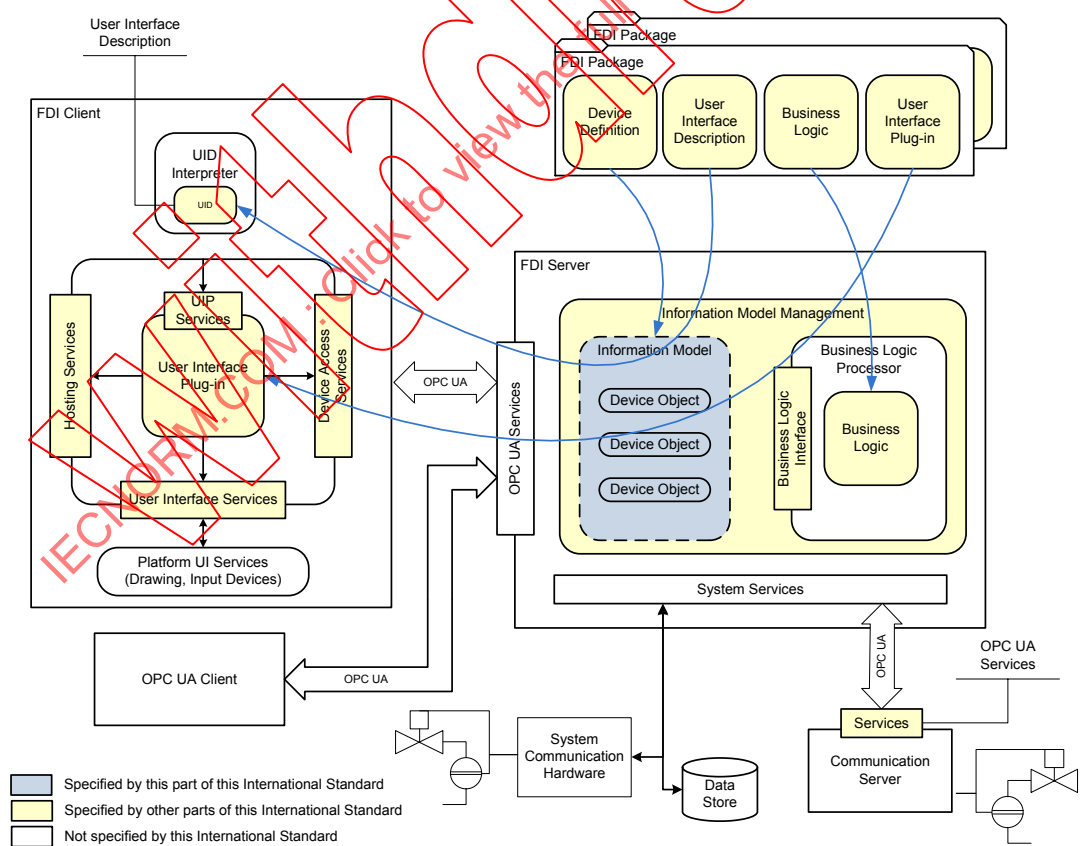


Figure 1 – FDI architecture diagram

2 Normative references

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

IEC 61784-1, *Industrial communication networks – Profiles – Part 1: Fieldbus profiles*

IEC 61804-3¹, *Function blocks (FB) for process control and Electronic Device Description Language (EDDL) – Part 3: EDDL syntax and semantics*

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

IEC 62541-4, *OPC unified architecture – Part 4: Services*

IEC 62541-5, *OPC unified architecture – Part 5: Information Model*

IEC 62541-6, *OPC unified architecture – Part 6: Mappings*

IEC 62541-8, *OPC unified architecture – Part 8: Data Access*

IEC 62541-100², *OPC unified architecture – Part 100: OPC UA for Devices*

IEC 62769-1, *Field Device Integration (FDI) – Part 1: Overview*

NOTE IEC 62769-1 is technically identical to FDI-2021

IEC 62769-2, *Field Device Integration (FDI) – Part 2: FDI Client*

NOTE IEC 62769-2 is technically identical to FDI-2022

IEC 62769-4, *Field Device Integration (FDI) – Part 4: FDI Packages*

NOTE IEC 62769-4 is technically identical to FDI-2024

IEC 62769-7, *Field Device Integration (FDI) – Part 7: FDI Communication Devices*

NOTE IEC 62769-7 is technically identical to FDI-2027

3 Terms, definitions, abbreviated terms, acronyms and conventions

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 62769-1 apply.

¹ To be published.

² Under consideration.

3.2 Abbreviated terms and acronyms

For the purposes of this document, the abbreviated terms and acronyms given in IEC 62769-1 as well as the following apply.

HMI	Human Machine Interface
SCADA	Supervisory Control and Data Acquisition
TCP	Transmission Control Protocol

3.3 Conventions for graphical notation

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 standard. NodeClasses representing types always have a shadow.

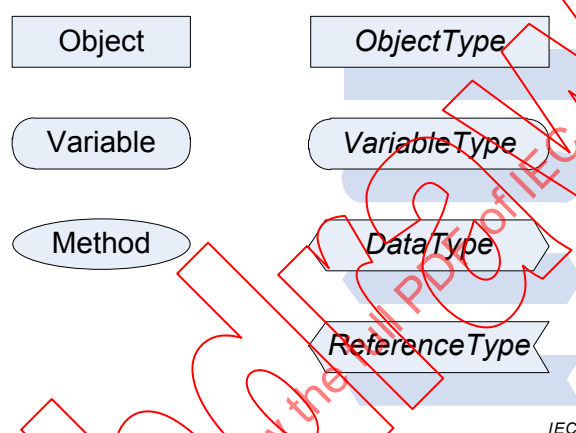
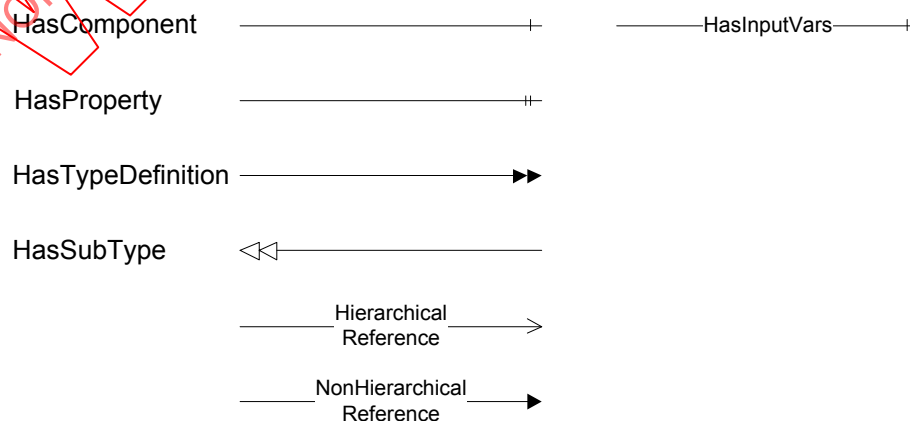


Figure 2 – OPC UA Graphical Notation for NodeClasses

Figure 3 shows the symbols for the ReferenceTypes used in this standard. The Reference symbol is normally pointing from the source Node to the target Node. The only exception is the HasSubType Reference. The most important References such as HasComponent, HasProperty, HasTypeDefinition and HasSubType have special symbols avoiding the name of the Reference. For other ReferenceTypes or derived ReferenceTypes the name of the ReferenceType is used together with the symbol.



IEC

Figure 3 – OPC UA Graphical Notation for References

Figure 4 shows a typical example for the use of the graphical notation. Object_A and Object_B are instances of the ObjectType_Y indicated by the HasTypeDefinition References. The ObjectType_Y is derived from ObjectType_X indicated by the HasSubType Reference. The Object_A has the components Variable_1, Variable_2 and Method_1.

To describe the components of an Object on the ObjectType the same NodeClasses and References are used on the Object and on the ObjectType such as for ObjectType_Y in the example. The Nodes used to describe an ObjectType are instance declaration Nodes.

To provide more detailed information for a Node, a subset or all Attributes and their values can be added to a graphical symbol (see for example Variable_1, the component of Object_A in Figure 4).

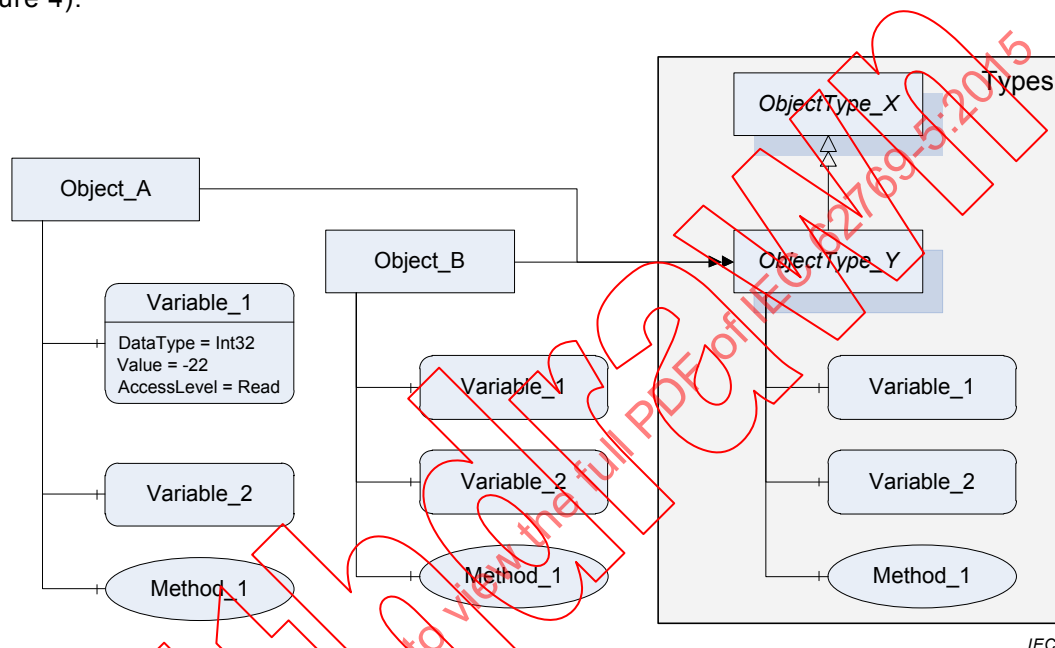


Figure 4 – OPC UA Graphical Notation Example

To improve readability, this document frequently includes the type name inside the instance box rather than displaying both boxes and a reference between them. This optimization is shown in Figure 5.

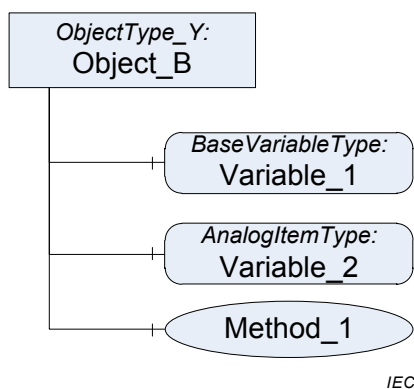


Figure 5 – Optimized Type Reference

4 Overview of OPC Unified Architecture

4.1 General

The main use case for OPC standards is the online data exchange between devices and HMI or SCADA systems. 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 provides functionality to browse through a hierarchical namespace containing data items and to read, write and monitor these items for data changes.

OPC UA incorporates features like Data Access, Alarms and Historical Data via platform independent communication mechanisms and generic, extensible and object-oriented modelling capabilities for the information a system wants to expose.

The current version of OPC UA defines an optimized binary TCP protocol for high performance intranet communication as well as a mapping to Web Services. The abstract service model does not depend on a specific protocol mapping and allows adding new protocols in the future. Features like security, access control and reliability are directly built into the transport mechanisms. Based on the platform independence of the protocols, OPC UA servers and clients can be directly integrated into devices and controllers.

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

The set of Objects and related information that an OPC UA Server makes available to Clients is referred to as its AddressSpace. The elements of the OPC UA Object Model are represented in the AddressSpace as a set of Nodes described by Attributes and interconnected by References. OPC UA defines various classes of Nodes to represent AddressSpace components most importantly Objects, Variables, Methods, ObjectTypes, DataTypes and ReferenceTypes. Each NodeClass has a defined set of Attributes.

Objects are used to represent components like folders, Devices or Networks. An Object is associated to a corresponding ObjectType that provides definitions for that Object.

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

Properties are Server-defined characteristics of Objects, DataVariables and other Nodes. Properties are not allowed to have Properties defined for them. An example for Properties of Objects is the Manufacturer Property of a Device.

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

4.2 Overview of OPC UA Devices

The OPC Unified Architecture for Devices (DI) (IEC 62541-100) standard is an extension of the overall OPC Unified Architecture standard series and defines information models associated with Devices. IEC 62541-100 describes three models which build upon each other as follows:

- The (base) Device Model is intended to provide a unified view of devices irrespective of the underlying device protocols.
- The Device Communication Model adds Network and Connection information elements so that communication topologies can be created.

- The Device Integration Host Model finally adds additional elements and rules required for host systems to manage integration for a complete system. It allows reflecting the topology of the automation system with the devices as well as the connecting communication networks.

The Devices information model specifies different ObjectTypes and other AddressSpace elements used to represent Devices and related components such as the communication infrastructure in an OPC UA AddressSpace. 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.

Figure 6 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 FunctionalGroupType are used to group the Parameters and Methods of the Device into logical groups. The FunctionalGroupType and the grouping concept are defined in IEC 62541-100 but the groups are DeviceType specific, i.e. the groups ProcessData and Configuration are defined by the TemperatureControllerType in this example.

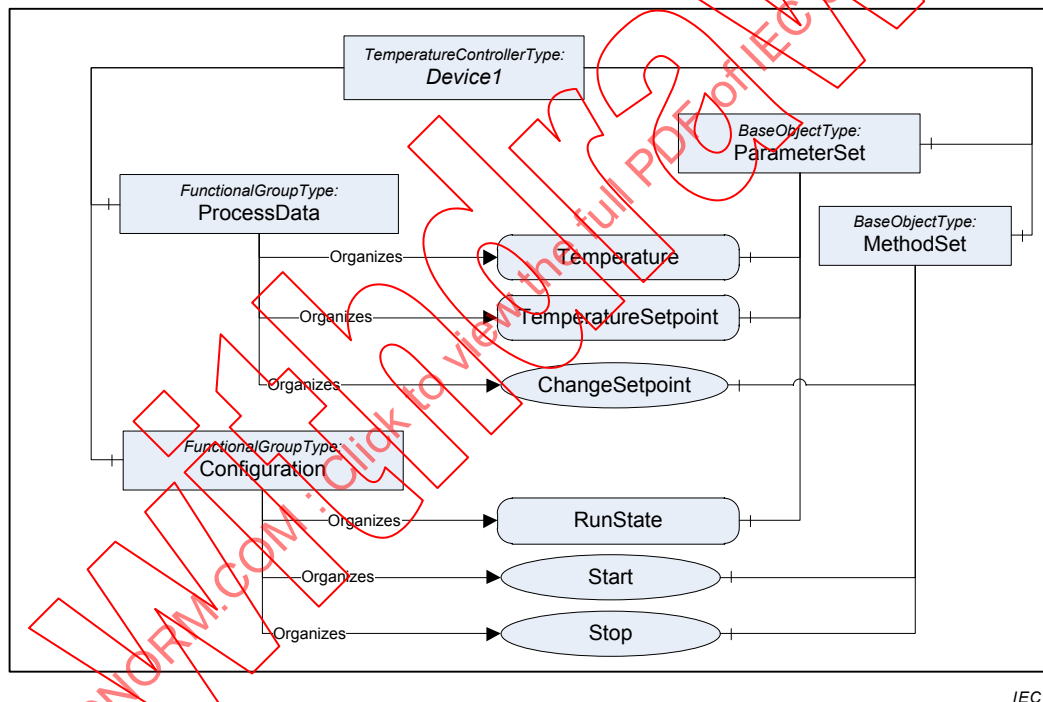


Figure 6 – OPC UA Devices Example: Functional Groups

Another IEC 62541-100 concept is illustrated in Figure 7. The ConfigurableObjectType is used to provide a way to group sub components of a Device and to indicate which types of sub components can be instantiated. The allowed types are referenced from the SupportedTypes folder. This information can be used by configuration clients to allow a user to select the type to instantiate as sub component of the Device.

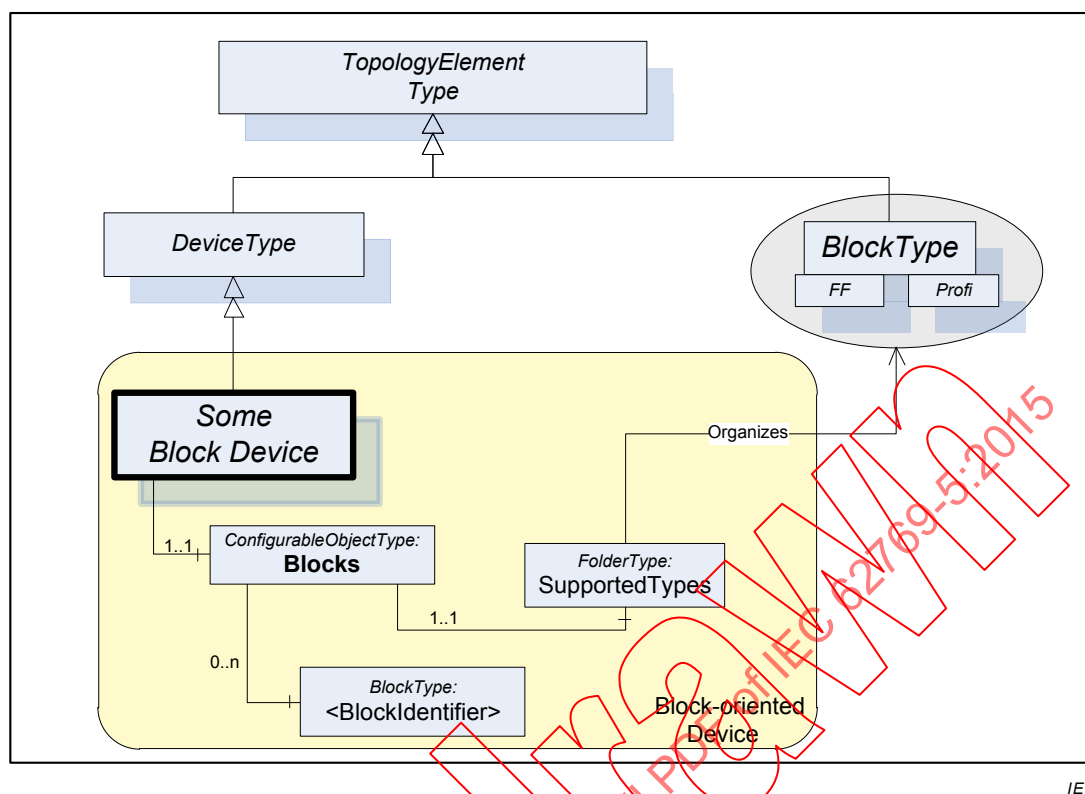


Figure 7 – OPC UA Devices example: Configurable components

The SupportedTypes folder can contain different subsets of ObjectTypes for different instances of the Block-oriented Device depending on their current configuration since the list contains only types that can be instantiated for the current configuration.

5 Concepts

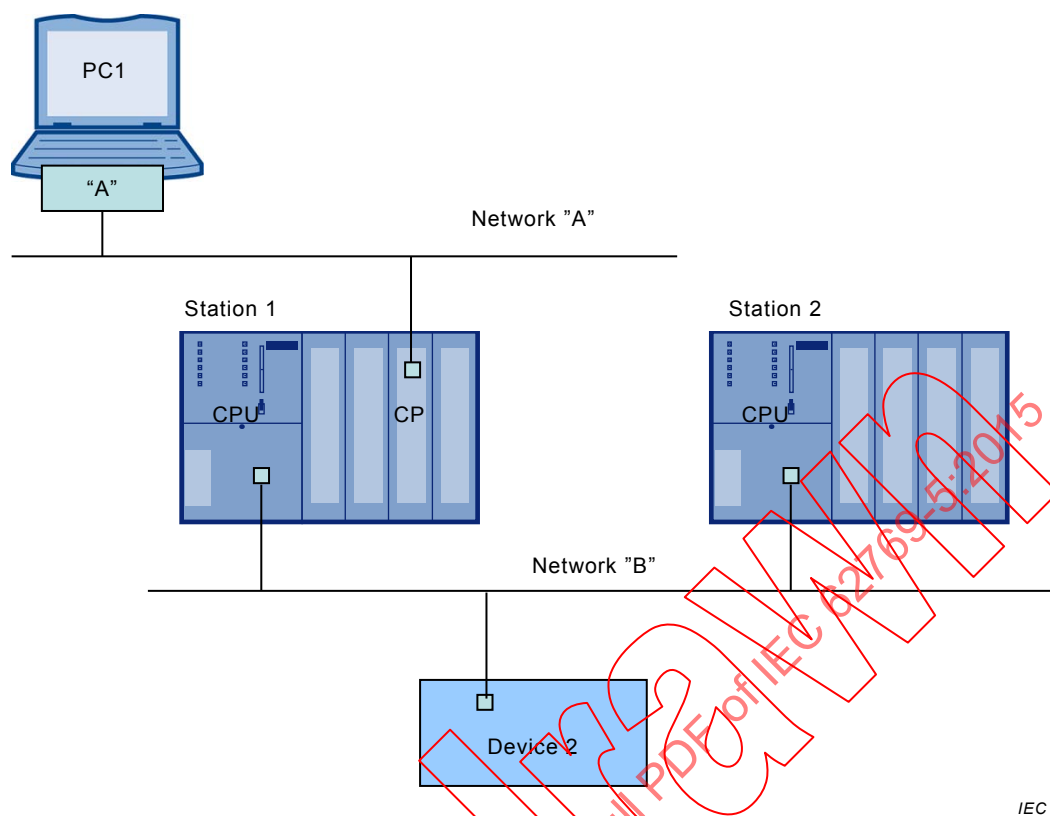
5.1 General

The FDI Server provides FDI Clients access to information about Device instances and Device types regardless of where the information is stored, for example, in the Device itself or in a data store. This information is provided via OPC UA Services and is called the FDI Information Model.

The FDI Information Model specifies the entities that may be accessed in the FDI Server, including their properties, relationships, and the operations that can be performed on them. Which types of Devices or other topological elements are available in a given FDI Server is driven largely by the information in the FDI Packages.

5.2 Device topology

One of the main tasks of the Information Model is to reflect the topology of the automation system. Therefore the Information Model represents the devices of the automation system as well as the connecting communication networks. The entry point Device Topology is the starting point within the Information Model for the topology of the automation system. The entry point Communication Devices contains the communication devices that are used by the FDI Server to access the elements of the topology. Figure 8 and Figure 9 illustrate an example configuration and the configured topology as it will appear in the FDI Server AddressSpace (details left out).



IEC

Figure 8 – Example of an automation system

The PC in Figure 8 represents the FDI Server box. The FDI Server communicates with devices connected to Network "A" via a Native Communication, and it communicates with devices connected to Network "B" via a Nested Communication.

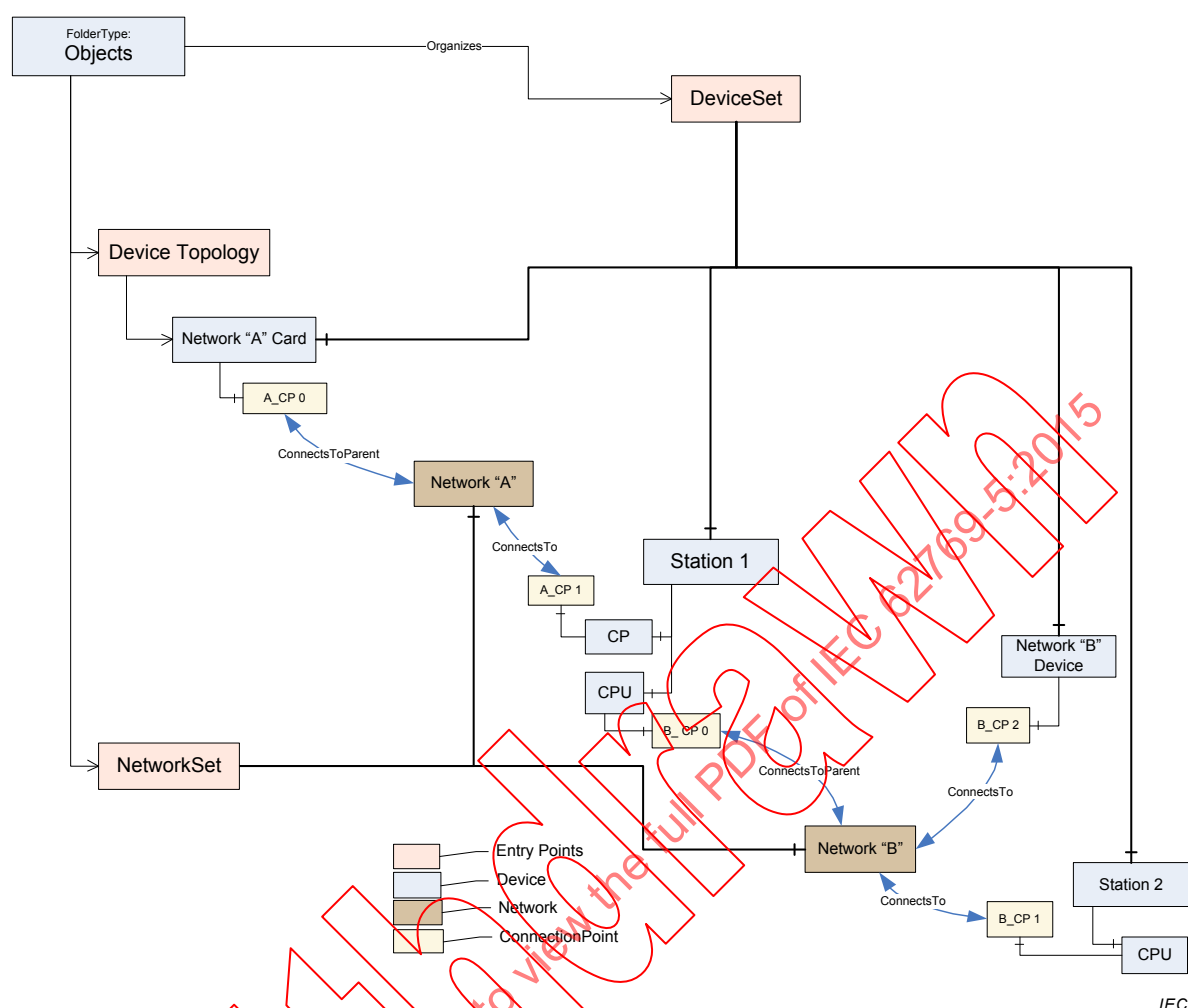


Figure 9 – Example of a Device topology

Coloured boxes are used to easily recognize the various types of information.

Brown boxes represent the networks. Light blue boxes represent the Devices and light yellow is used for Connection Points.

Light pink boxes represent the entry points that assure common behaviour across different implementations:

- DeviceTopology: Starting node for the topology configuration.
- DeviceSet: All instantiated Devices are components of this Object, i.e., they exist in the AddressSpace independently of the Device Topology.
- NetworkSet: All Networks are components of this Object.

5.3 Online/offline

Management of the Device Topology is a configuration task, i.e., the elements in the topology (Devices, Networks, and Connection Points) are usually configured “offline” and – at a later time – will be validated against their physical representative in a real network.

To support explicit access to either the online or the offline information, each element is represented by two instances that are schematically identical, i.e., there exists a ParameterSet, FunctionalGroups, and so on. A Reference connects the online and offline representation and allows navigating between them.

5.4 Catalogue (Type Definitions)

The supported (sub-types of) `TopologyElements` are organised as Type definitions in the OPC UA AddressSpace forming some kind of a catalogue. These definitions typically are generated based on descriptive information from FDI Packages. The Type definitions contain the Parameters, and default values for Parameters, Methods, Actions and Functional Groups including user interface elements. The FDI Server can include folders in the type model to organise the types according to manufacturer or other criteria.

Type definitions can then be used to create instances of Devices in the OPC UA AddressSpace. Instances can be created either offline or based on data determined by Scanning. Figure 10 illustrates an example of some Type definitions (details left out) as they may exist in the AddressSpace.

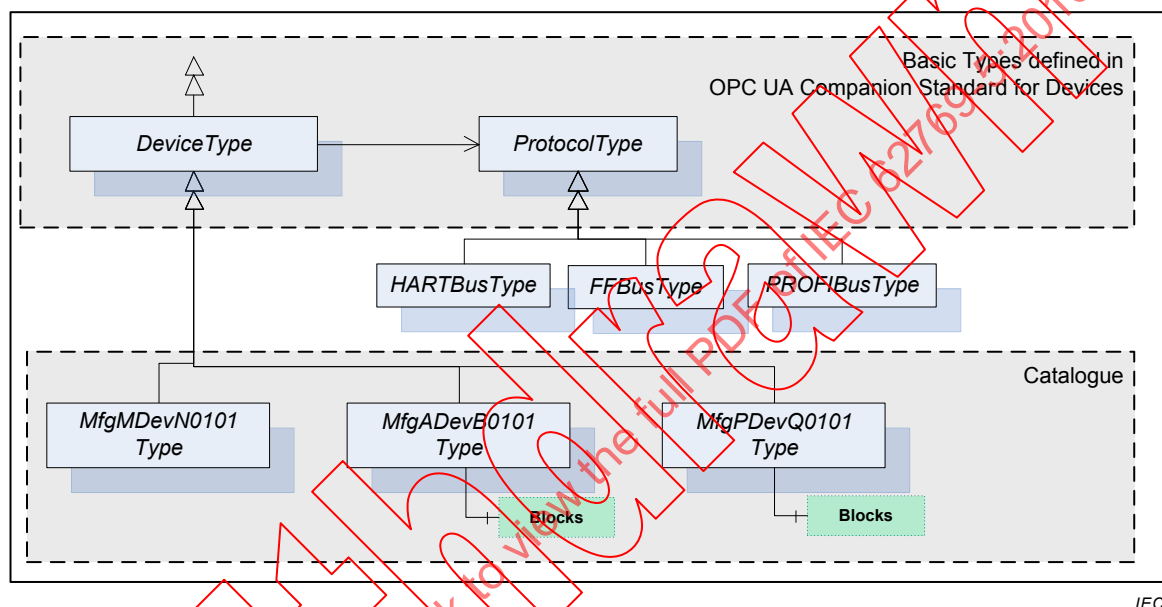


Figure 10 – Example Device Types representing a catalogue

5.5 Communication

In order to integrate Devices, the FDI Server needs to be able to communicate to them. This can be done using Native Communication or Nested Communication.

The example in Figure 9 above for instance specifies that the FDI Server has direct access to the PROFINET Network using its PROFINET network card. In order to access "Station 2", the FDI Server has to go through Station 1, which provides the communication services for the PROFIBUS DP Network (see IEC 61784-1, CPF 3). This can be achieved through Nested Communication, which is specified in IEC 62769-4. Communication Devices and Communication Servers are specified in IEC 62769-7.

6 AddressSpace organization

To promote interoperability of FDI Clients and FDI Servers, a set of Objects and relationships are defined in the following subclauses. FDI Servers can implement a subset of these standard Nodes, depending on their capabilities.

Based on OPC UA rules, an OPC UA Server separates the AddressSpace in two parts:

- The Types-part contains information about all components that have been generated based on descriptive information from FDI Packages (see 5.4).

- b) The Objects-part contains the Device Topology with all instantiated components. All instances of the AddressSpace are related to a type of the Types folder
- The entry points DeviceSet, NetworkSet, and DeviceTopology are formally defined in IEC 62541-100.
 - DeviceTopology is used to aggregate the top level Networks that provide access to all instances that constitute the Device Topology ((sub-)networks, devices and communication elements). Some example elements are shown here and highlighted using the green colour.
 - All instantiated Devices are components of the DeviceSet Object, i.e., they exist in the AddressSpace independent of the Device Topology. All Networks are components of the NetworkSet Object.

FDI Servers can either automatically create Device Objects or they may only show the available types (SupportedTypes folder) and leave it to the user to create proper instances. When using native communication the system will typically provide the Device Topology without having to have it configured by the FDI Client.

7 Device Model for FDI

7.1 General

As mentioned above, IEC 62541-100 specifies the fundamental types needed for FDI like the TopologyElementType, the DeviceType and the ProtocolType (the fieldbus protocol). Clause 7 briefly repeats important design elements specified in IEC 62541-100 and specifies additional types that are not in IEC 62541-100.

7.2 Online/offline

All elements that appear in the Device Topology (Devices, Networks, and Connection Points) including their relationship correspond to information stored in the FDI Server's configuration database. Management of these elements most commonly requires access to the physical component/device (called online data in this standard) and also the storage and administration of related data in a configuration database (called offline data).

To support explicit access to either the online or the offline information, each element is represented by two instances that are schematically identical, i.e., there exists a ParameterSet, FunctionalGroups, and so on. A Reference connects online and offline representation and allows to navigate between them. This is illustrated in Figure 11.

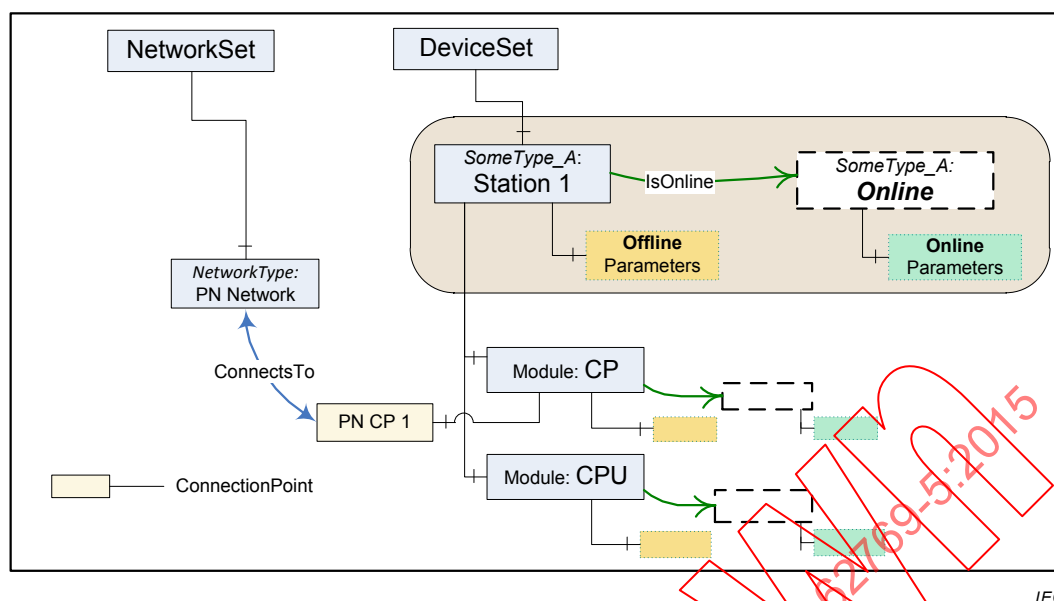


Figure 11 – Online component for access to device data

Support of online/offline is mandatory for FDI Servers. Detailed information of the model and the formal definitions are specified in IEC 62541-100.

7.3 Device health

The DeviceHealth Property indicates the status of a device as defined by NAMUR NE107. FDI Clients can read or monitor this Property to determine the device condition.

Servers determine the health status using the EDD METHOD GetHealthStatus defined in IEC 62769-4. The frequency at which Servers actually examine the health status may vary from several seconds up to minutes.

Support of the DeviceHealth Property is mandatory for Device Objects. Detailed information of the model and the formal definition are in IEC 62541-100.

7.4 User interface elements

7.4.1 General

IEC 62541-100 defines in an abstract way, how Servers can expose user interface elements for Clients to display a user interface specific to a FunctionalGroup of a TopologyElement.

Subclause 7.4 specifies two concrete user element types: descriptive user interface elements (UIDs) and programmed (executable) user interface elements (UIPs). UIPs are never referenced directly from a FunctionalGroup. They are always indirectly referenced from a UID by means of their UipId.

Figure 12 illustrates the type hierarchy of the user interface elements defined in IEC 62541-100 and in this standard.

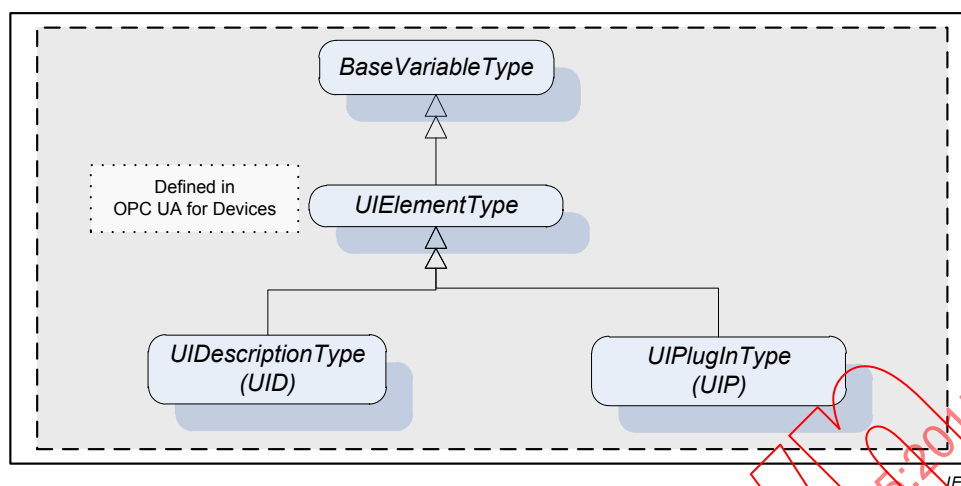


Figure 12 – Hierarchy of user interface Types

7.4.2 UI Description Type

FDI Servers may provide a descriptive user interface element (a UID) for each FunctionalGroup. Such an element will be rendered by the FDI Client. The UIDescriptionType is formally specified in Table 1.

Table 1 – UIDescriptionType Definition

Attribute	Value				
BrowseName	UIDescriptionType				
DataType	String				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherits the Properties of the UIElementType defined in IEC 62541-100.					

The Value Attribute provides the UID as a String containing an XML Element. See IEC 62769-2 for the syntax of UID elements. The XML Schema for the UID Value of all exposed devices adheres to the same FDI Technology Version as indicated by the FDIServerVersion Property (see Clause 14).

7.4.3 UI Plug-in Type

A User Interface Plug-in (UIP) is a software module that is hosted and run by an FDI Client. In contrast to a User Interface Description (UID) it is an executable UI element.

Details on hosting and running Plug-ins are specified in IEC 62769-2. The UIPlugInType is formally specified in Table 2.

Table 2 – UIPlugInType Definition

Attribute	Value				
BrowseName	UIPlugInType				
DataType	Byte				
ValueRank	1 – one dimensional array				
ArrayDimensions	UInt32[1] – the length (number of bytes) of the array				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherits the Properties of the UIElementType defined in IEC 62541-100.					
HasProperty	Variable	UIPVariantVersion	String	PropertyType	Mandatory
HasProperty	Variable	FDITechnologyVersion	String	PropertyType	Mandatory
HasProperty	Variable	RuntimeId	String	PropertyType	Mandatory
HasProperty	Variable	CpuInformation	String	PropertyType	Mandatory
HasProperty	Variable	PlatformId	String	PropertyType	Mandatory
HasProperty	Variable	Style	String	PropertyType	Mandatory
HasProperty	Variable	StartElementName	String	PropertyType	Mandatory
HasComponent	Object	Documentation		FolderType	Optional

UIPs may exist in multiple variants for different platforms or supporting different versions. The UipId (a unique identifier defined in the FDI package) identifies the UIP, not a specific variant.

UIPs do not have to be exposed in the AddressSpace. With the UipId the FDI Client can retrieve the NodeIds of UIP Variants and their Properties by calling the OPC UA TranslateBrowsePathToNodeIds Service with the Device NodeId as the startingNode and the following list of relative names:

- "UIPSet/<UipId>"
- "UIPSet/<UipId>/RuntimeId"
- "UIPSet/<UipId>/CpuInformation"
- "UIPSet/<UipId>/PlatformId"
- "UIPSet/<UipId>/FDITechnologyVersion"
- "UIPSet/<UipId>/Style"
- "UIPSet/<UipId>/StartElementName"
- "UIPSet/<UipId>/UIPVariantVersion"

NOTE "UIPSet" is an identifier for the Server and does not have to be a Node in the AddressSpace.

The FDI Server returns arrays of NodeIds for each relative name. The number of entries in each array matches the number of UIP Variants for the UipId. The FDI Client can read the property values using the received NodeIds and choose the appropriate UIP Variant based on FDITechnologyVersion, RuntimeId, CpuInformation, and PlatformId.

The Value Attribute provides the UIP executable. The exact representation is technology dependent (see IEC 62769-6). The ArrayDimensions Attribute shall specify the size (number of bytes) of the UIP.

FDI Clients need to be able to handle large UIPs. Reading large UIPs with a single Read operation may not be possible due to configured limits in either the FDI Client or the FDI Server stack. The default maximum size for an array of bytes is 1 MegaByte. FDI Clients can use the IndexRange in the OPC UA Read Service (see IEC 62541-4) to read a UIP in – for instance – one megabyte chunks. It is up to the FDI Client whether it starts without index and repeats with an indexRange only after an error or whether it always uses an indexRange.

The following Properties help the FDI Client to identify which UIP fits best to its environment:

- **UIPVariantVersion:**
The version of this UIP Variant.
- **FDITechnologyVersion:**
FDI Technology Version according to which the UIP is developed. A UIP shall always be capable of running in a client/server system with the same major version and different minor/maintenance version.
- **RuntimeId:**
Runtime environment of the UIP as specified in IEC 62769-6.
- **CpuInformation:**
Provides additional information about the execution environment associated with the RuntimeId. The allowed values are specified in IEC 62769-6.
- **PlatformId** defines the type of platform on which this UIP Variant is supported. An FDI Client can choose a particular UIP Variant if it matches the FDI Client's platform (see IEC 62769-4 for the concrete definitions).
 - “Workstation”– with regular screen resolution capabilities, memory capabilities, input devices available (like mouse and keyboard)
 - “Mobile”– limited screen resolution, memory and input devices possible
- **Style:**
Defines whether the UIP shall be run “modal” or “modeless” as defined in IEC 62541-4. Currently, the values “Dialog” and “Window” are defined. While “Dialog” requires a modal window, a UIP with style “Window” will be invoked either in a modal or a non-modal window as defined in IEC 62769-2.
- **StartElementName:**
Element needed to start this UIP Variant. IEC 62769-6 specifies how this information is used when activating the UIP.

Documents provided for a UIP Variant are exposed as Variables organized in the Documentation folder. In most cases they will represent a product manual, which can exist as a set of individual documents. The information can be retrieved by reading the Variable value which is represented as a ByteString. The complete ByteString shall be interpreted as a PDF file. FDI Clients need to be aware that the contents that these variables represent may be large. Reading large values with a single Read operation may not be possible due to configured limits in either the FDI Client or the FDI Server stack. The default maximum size for an array of bytes is 1 MegaByte. It is recommended that FDI Clients use the IndexRange in the OPC UA Read Service (see IEC 62541-4) to read these Variables in chunks, for example, one megabyte chunks.

7.5 Type-specific support information

Each DeviceType may have a set of additional data. These are mainly images, documents, or protocol-specific data. The various types of information are organized into different folders.

See IEC 62541-100 for the formal definition of support information.

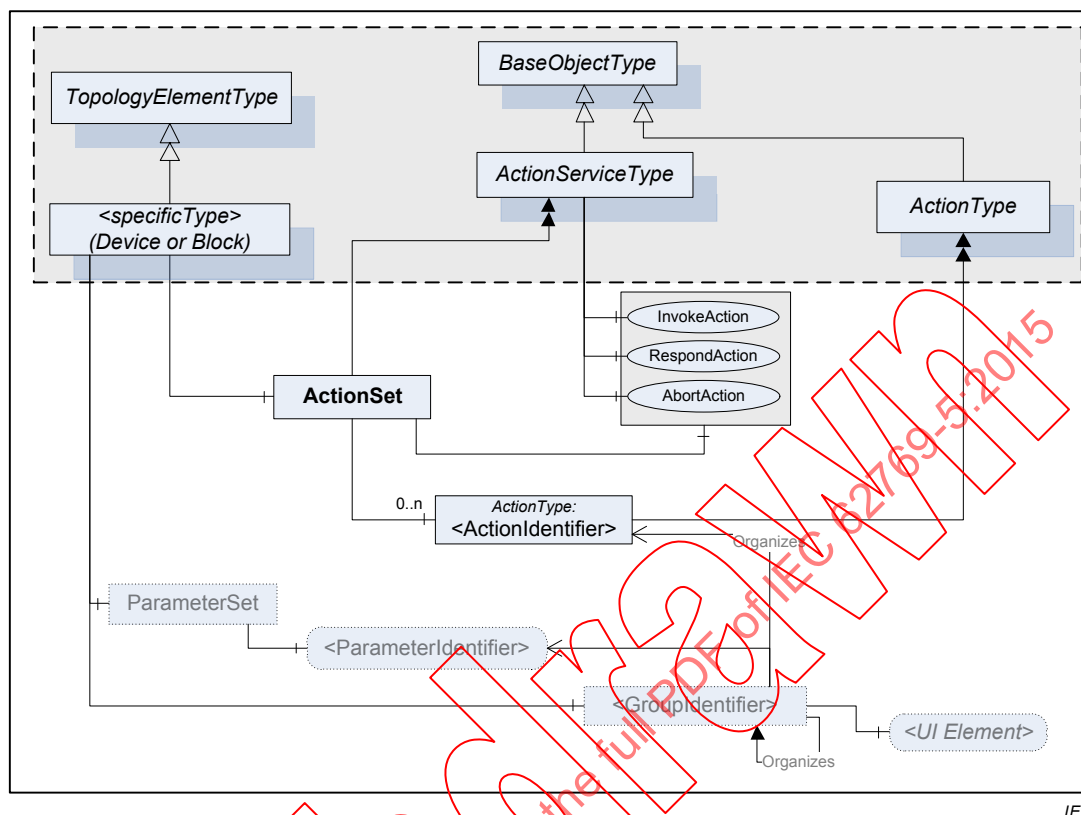
7.6 Actions

7.6.1 Overview

Actions are operations that are executed in the FDI Server on behalf of a topology element. Once invoked with the InvokeAction Method, Actions can make various state transitions until completed. The actual state is accessible via a transient, non-browsable Variable the NodeId of which is returned by the InvokeAction Method.

FDI Clients can subscribe to this Variable to receive updates concerning the Action execution (Action data). Action data may report a state transition or a request to the FDI Client that input is necessary for continuation. The FDI Client can resume the execution with the RespondAction Method and submit the requested data.

Figure 13 illustrates how Actions are integrated into a TopologyElement. It is formally defined in Table 3.



IEC

Figure 13 – Integration of Actions within a TopologyElement

Table 3 – TopologyElementType with additions for Actions

Attribute	Value				
BrowseName	TopologyElementType				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherits the Properties of the BaseObjectType defined in [IEC 62541-5]					
HasComponent	Object	ParameterSet		BaseObjectType	Optional
HasComponent	Object	MethodSet		BaseObjectType	Optional
HasComponent	Object	<GroupIdentifier>		FunctionalGroupType	OptionalPlaceHolder
HasComponent	Object	Identification		FunctionalGroupType	Optional
HasComponent	Object	Lock		LockingServicesType	Optional
HasComponent	Object	ActionSet		ActionServiceType	Optional

ActionSet is used to aggregate the Actions for a specific TopologyElement. It is added to the TopologyElementType so that it can be used on any sub-type as well. This Object is only available in an instance if Actions exist for this TopologyElement.

Action related extensions to the FunctionalGroup ObjectType are formally defined in Table 4.

Table 4 – FunctionalGroupType with additions for Actions

Attribute	Value				
BrowseName	FunctionalGroupType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Inherits the Properties of the FolderType defined in [IEC 62541-5]					
HasComponent	Object	<GroupIdentifier>		FunctionalGroupType	OptionalPlaceHolder
Organizes	Variable	<ParameterIdentifier>		BaseDataVariableType	OptionalPlaceHolder
Organizes	Method	<MethodIdentifier>			OptionalPlaceHolder
HasComponent	Variable	UIElement	BaseDataType	UIElementType	Optional
Organizes	Object	<ActionIdentifier>		ActionType	Optional

<ActionIdentifier> refers to Actions available and exposed for this FunctionalGroup.

7.6.2 Action Type

This ObjectType defines the structure of an Action. It is formally defined in Table 5.

Table 5 – ActionType Definition

Attribute	Value				
BrowseName	ActionType				
IsAbstract	True				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the BaseObjectType is defined in IEC 62541-5.					

The FDI Client determines the required invocation arguments for an Action from the UID.

7.6.3 ActionService Type

The ActionServiceType defines the Methods to invoke and control Actions. Instances of this type aggregate the Actions for a specific topology element. The ActionServiceType is formally defined in Table 6. Its use is illustrated in Figure 13.

Table 6 – ActionServiceType Definition

Attribute	Value				
BrowseName	ActionServiceType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the BaseObjectType defined in IEC 62541-5.					
HasComponent	Method	InvokeAction			Mandatory
HasComponent	Method	RespondAction			Mandatory
HasComponent	Method	AbortAction			Mandatory
HasComponent	Object	<ActionIdentifier>		ActionType	Optional

The ActionServiceType and each instance of this Type share the same Methods. The NodeId of these Methods will be fixed and defined in this standard. FDI Clients therefore do not have to browse for these Methods. They can use the fixed NodeId as the MethodId of the Call Service.

The OPC UA StatusCode Bad_MethodInvalid shall be returned from the Call Service for elements where the ActionService Methods are not supported.

<ActionIdentifier> stands for one or several Actions. Actions (Objects of ActionType) exist only in instances of the ActionServiceType, i.e., when an instance of this ObjectType is added to a TopologyElement. No Actions (Objects of ActionType) exist in the ActionServiceType itself.

7.6.4 ActionService Object

The support of the ActionService for an Object is declared by aggregating an instance of the ActionService Type as illustrated in Figure 14.

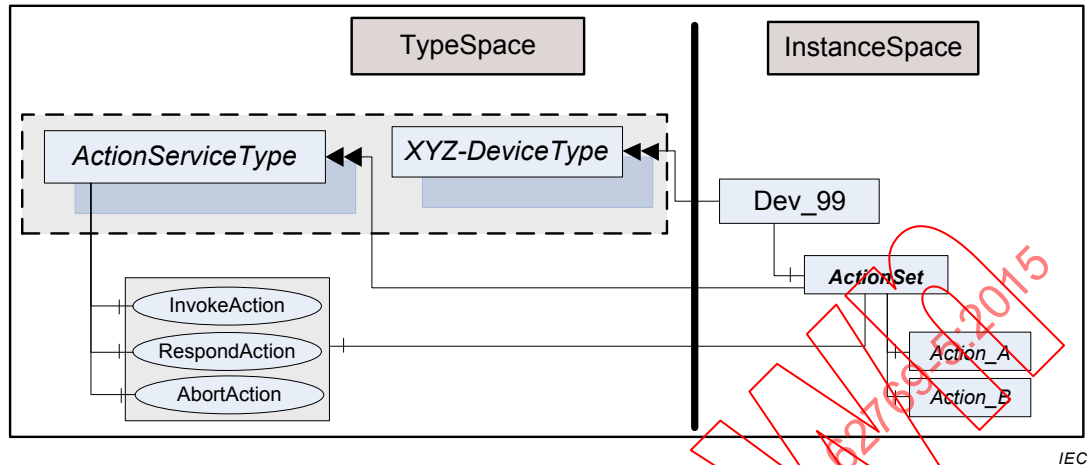


Figure 14 – Action Service

This Object is used as container for the ActionService Methods and shall have the BrowseName ActionSet. It is formally defined in Table 28. HasComponent is used to reference from a TopologyElement (for example, a Device) to its “ActionService” Object.

The ActionServiceType and each ActionSet Object share the same Methods. Actions will typically be shared by all instances of the same Device Type.

7.6.5 InvokeAction Method

InvokeAction is used to start an Action. It immediately returns after the state machine has been created. The “ActionSet” component of the TopologyElement (Device) on behalf of which the Action shall be invoked is specified via the ObjectId argument of the Call Service.

Explicit locking is required. If the Device has not been locked, the FDI Server will reject the request.

After InvokeAction returns, the FDI Client shall subscribe to the Value Attribute of the ActionNodeId. The Value Attribute contains an XML element (DataType = String)) that reflects the current state of the Action as well as additional data (depending on state). The FDI Client therefore will get a DataChange notification whenever the state of the Action changes.

See IEC 62769-2 for the Action state diagrams as well as the XML Schema of the ActionNodeId value.

FDI Servers shall cache Action state data for an appropriate period of time (a few seconds) so that no state information is lost until the FDI Client had a chance to subscribe.

The signature of this Method is specified below. Table 7 and Table 8 specify the arguments and AddressSpace representation, respectively.

Signature

```
InvokeAction (
    [in] String      ActionName,
    [in] String      MethodArguments,
```

```

[out] NodeId      ActionNodeId,
[out] Int32       InvokeActionError);

```

Table 7 – InvokeAction Method Arguments

Argument	Description
ActionName	String portion of the BrowseName of the Action as used in the ActionServiceType instance.
MethodArguments	XML document that contains the Action input arguments (if any). The ListOfActionArguments XML Schema defined in IEC 62769-2 is used for the MethodArguments parameter.
ActionNodeId	Non-browsable node of type Variable. This node is used both to identify the instance of the Action state machine and for access to the Action state information.
InvokeActionError	0 – OK. -1 - E_LockRequired – the element is not locked as required -2 - E_UnknownAction – the passed name is not a valid action for this element.

Table 8 – InvokeAction Method AddressSpace Definition

Attribute	Value				
BrowseName	InvokeAction				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Mandatory

7.6.6 RespondAction Method

RespondAction is used by the FDI Client to provide the requested value or selection of a UI function request (if applicable). The “ActionSet” component of the TopologyElement on behalf of which the Action had been invoked is specified via the ObjectId argument of the Call Service.

The signature of this Method is specified below. Table 9 and Table 10 specify the arguments and AddressSpace representation, respectively.

Signature

```

RespondAction (
    [in] NodeId      ActionNodeId,
    [in] String      Response,
    [out] Int32      RespondActionError);

```

Table 9 – RespondAction Method Arguments

Argument	Description
ActionNodeId	NodeId of a transient Variable that represents and identifies the executing Action. This Id is returned by the InvokeAction Method.
Response	XML document that contains the response (value or selection). See IEC 62769-2 for the Xml Schema for the Response parameter.
RespondActionError	0 – OK -1 – E_InvalidAction – the NodeId does not refer to an existing action -2 – E_InvalidResponse – the passed response data could not be interpreted

Table 10 – RespondAction Method AddressSpace Definition

Attribute	Value				
BrowseName	RespondAction				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Mandatory

7.6.7 AbortAction Method

AbortAction is used by the FDI Client to abort an Action execution. This Method call immediately returns. The FDI Client is informed via a notification event on the ActionNodeId Variable when the Action enters the Aborting state.

The “ActionSet” component of the TopologyElement on behalf of which the Action had been invoked is specified via the ObjectId argument of the Call Service.

The signature of this Method is specified below. Table 11 and Table 12 specify the arguments and AddressSpace representation, respectively.

Signature

```

AbortAction (
    [in]  NodeId          ActionNodeId,
    [out] Int32           AbortActionError)

```

Table 11 – AbortAction Method Arguments

Argument	Description
ActionNodeId	NodeId of a transient Variable that represents and identifies the executing Action. This Id is returned by the InvokeAction Method.
AbortActionError	Q – OK -1 – E_InvalidAction – the NodeId does not refer to an existing action

Table 12 – AbortAction Method AddressSpace Definition

Attribute	Value				
BrowseName	AbortAction				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Mandatory

8 Network and connectivity

Network and connection information elements are required to create communication topologies.

A Network represents the communication means for Devices that are connected to it. It includes wired and wireless technologies. ConnectionPoints represent the interface (interface card) of a Device to a Network. A specific sub-type shall be defined for each protocol.

These elements are described and formally defined in IEC 62541-100.

9 Utility functions

9.1 Overview

Clause 9 provides specific services for certain FDI information elements.

9.2 Locking

Locking is the means to avoid concurrent modifications to a Device or Network and their components. FDI Clients shall use the Locking Services for any changes (e.g., write operations and Action invocations).

The main purpose of locking a Device is avoiding concurrent device modifications. The main purpose of locking a Network is avoiding concurrent topology changes.

When locking a Device, the lock always applies to both the online and the offline version.

When locking a Modular Device, the lock applies to the complete device (including all modules). Equally, when locking a Block Device, the lock applies to the complete device (including all blocks).

If no lock is applied to the top-level Device (for Modular Device or for Block Device), the sub-devices or blocks, respectively, can be locked independently.

When locking a Network, the lock applies to the Network and all connected Devices. If any of the connected Devices provides access to a sub-ordinate Network (such as a Gateway), the sub-ordinate Network and its connected Devices are locked as well.

The LockingService is fully described and formally defined in IEC 62541-100.

9.3 EditContext

9.3.1 Overview

An EditContext can be used to make changes to Variable values visible to the Server without applying them to the Device. The FDI Server provides the EditContext concept to support Clients in their editing task.

The EditContext is specified in IEC 62541-3. Following is the OPC UA Information Model including the Methods to maintain EditContext instances.

EditContext is exposed as an AddIn capability which is comparable to the interface technology found in some programming languages. The EditContext service is modelled as an ObjectType and instances of this type are added to the Device with a pre-defined BrowseName. Additional AddIn examples are defined in IEC 62541-100.

When reading or subscribing to Variable values registered in an EditContext, the following FDI-specific StatusCodes may occur (see Clause 11):

- Good_Edited distinguishes values that have been edited but have not been written to the Device.
- Uncertain_DominantValueChanged indicates that dependent values are invalid and will be recalculated after the dominant value has been applied to the device. In the offline case the dependent value has to be written by the Client as well.
- Good_DependentValueChanged indicates that a dependent value has been changed but the change has not been applied to the device.

9.3.2 EditContext Type

The EditContextType comprises the EditContext Methods. It is formally defined in Table 13.

Table 13 – EditContextType Definition

Attribute	Value				
BrowseName	EditContextType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the BaseObjectType defined in IEC 62541-5.					
HasComponent	Method	GetEditContext			Mandatory
HasComponent	Method	RegisterNodesByIid			Mandatory
HasComponent	Method	RegisterNodesByRelativePath			Mandatory
HasComponent	Method	Apply			Mandatory
HasComponent	Method	Reset			Mandatory
HasComponent	Method	Discard			Mandatory

The StatusCode Bad_MethodInvalid shall be returned from the Call Service for elements where the EditContext Methods are not supported. Bad_UserAccessDenied shall be returned if the Client User does not have the permission to call the Methods.

9.3.3 EditContext Object

The support of EditContext for an Object is declared by aggregating an instance of the EditContextType as illustrated in Figure 15.

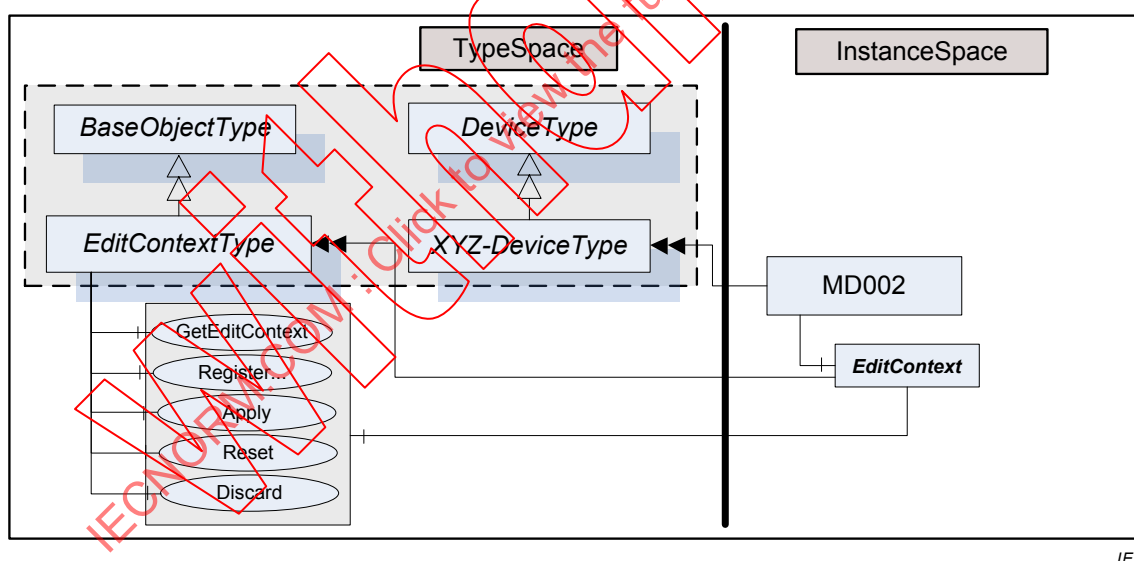


Figure 15 – EditContext type and instance

This Object is used as container for the EditContext Methods and shall have the BrowseName EditContext. HasComponent is used to reference from a Device to its “EditContext” Object.

The EditContextType and each instance may share the same Methods.

9.3.4 GetEditContext Method

Returns an EditContext as specified in IEC 62541-3.

The signature of this Method is specified below. Table 14 and Table 15 specify the arguments and AddressSpace representation, respectively.

Signature

```

GetEditContext (
    [in] String      ParentId,
    [in] WindowMode TargetWindowMode,
    [out] String     EditContextId,
    [out] Int32      GetEditContextStatus);

```

Table 14 – GetEditContext Method Arguments

Argument	Description
ParentId	If Null, a root instance is requested. Otherwise, the Client passes the identifier of a previously acquired EditContext, indicating that it will create a sub-window.
TargetWindowMode	An enumeration that indicates the User Interface element used for this context. 1 – Modal Window 2 – NonModal Window 3 – UIP
EditContextId	A string identifier created by the Server.
GetEditContextStatus	0 – OK -1 – E_NotSupported – the element does not support the EditContext Service

Table 15 – GetEditContext Method AddressSpace Definition

Attribute	Value				
BrowseName	GetEditContext				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Mandatory

9.3.5 RegisterNodes Method

This Method is used to register Nodes with an EditContext. It returns new NodeIds that have to be used to address this Node within the EditContext.

The signature of this Method is specified below. Table 16 and Table 17 specify the arguments and AddressSpace representation, respectively.

Signatures

```

RegisterNodes (
    [in] String      EditContextId,
    [in] RegistrationParameters[] NodesToRegister,
    [out] RegisterNodesResult      RegisterNodesStatus);

```

Table 16 – RegisterNodes Method Arguments

Argument	Description
EditContextId	Identifier of an EditContext that was previously acquired with a GetEditContext call.
NodesToRegister	An array of structures for each node to register.
RegisterNodesStatus	A structure with overall execution status and result data for each Node to register.

Table 17 – RegisterNodes Method AddressSpace Definition

Attribute	Value				
BrowseName	RegisterNodes				
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Mandatory

The RegistrationParameters DataType defines a single Node to be registered. Its elements are defined in Table 19.

Table 18 – RegistrationParameters DataType Structure

Name	Type	Description										
RegistrationParameters	structure	This structure specifies one of the nodes to register.										
path	RelativePath	RelativePath for this Node to register. The RelativePath type is defined in IEC 62541-4.										
selectionFlags	UInt32	A bit mask that identifies the EditContext-specific NodeIds to be returned in the RegisterNodesResult structure. The value of this parameter shall contain at least one of the following values. No value will be rejected with E_InvalidSelectionFlags. <table><tr><th>Bit Value</th><th>NodeId to return</th></tr><tr><td>0x0000 0001</td><td>Online / ContextNodeId</td></tr><tr><td>0x0000 0002</td><td>Online / DeviceNodeId</td></tr><tr><td>0x0000 0004</td><td>Offline / ContextNodeId</td></tr><tr><td>0x0000 0008</td><td>Offline / DeviceNodeId</td></tr></table>	Bit Value	NodeId to return	0x0000 0001	Online / ContextNodeId	0x0000 0002	Online / DeviceNodeId	0x0000 0004	Offline / ContextNodeId	0x0000 0008	Offline / DeviceNodeId
Bit Value	NodeId to return											
0x0000 0001	Online / ContextNodeId											
0x0000 0002	Online / DeviceNodeId											
0x0000 0004	Offline / ContextNodeId											
0x0000 0008	Offline / DeviceNodeId											

The RegisterNodesResult DataType includes overall status information and result data for each Node to be registered. Its elements are defined in Table 19.

Table 19 – RegisterNodesResult DataType Structure

Name	Type	Description
RegisterNodesResult	structure	This structure specifies the registration result.
status	Int32	0 – OK – the field registeredNodes contains a result for each Node to register -1 – E_InvalidId – the specified EditContext is unknown the registeredNodes field is empty
registeredNodes	RegisteredNode[]	The list contains EditContext-specific NodeIds for the registered Node. The Client has to use these NodeIds for all subsequent OPC UA Service calls
nodeStatus	Int32	0 – OK -1 – E_InvalidNode – an invalid Node has been registered. See IEC 62541-3 for details. -2 – E_InvalidSelectionFlags – the RegistrationParameters for this Node contained no value
onlineContextNodeId	NodeId	This NodeId shall be used to address the online representation of the Node in the EditContext. See IEC 62541-3 for details.
onlineDeviceNodeId	NodeId	This NodeId shall be used to address the online representation of the Node in the Device. See IEC 62541-3 for details.
offlineContextNodeId	NodeId	This NodeId shall be used to address the online representation of the Node in the EditContext. See IEC 62541-3 for details.
offlineDeviceNodeId	NodeId	This NodeId shall be used to address the online representation of the Node in the Device. See IEC 62541-3 for details.

9.3.6 Apply Method

This Method is used to apply values that have been modified in the EditContext.

The signature of this Method is specified below. Table 20 and Table 21 specify the arguments and AddressSpace representation, respectively.

Signature

```

Apply (
    [in] String      EditContextId,
    [out] ApplyResult ApplyStatus);

```

Table 20 – Apply Method Arguments

Argument	Description
EditContextId	Identifier of an EditContext that was previously acquired with a GetEditContext call.
ApplyStatus	A structure with overall execution status and status information for each transferred Variable.

Table 21 – Apply Method AddressSpace Definition

Attribute	Value				
BrowseName	Apply				
References	NodeClass	BrowseName	Data Type	Type Definition	Modelling Rule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Mandatory

The ApplyResult DataType includes overall Apply status information and status information for each Variable that could not be transmitted. Its elements are defined in Table 22 and Table 19.

Table 22 – ApplyResult DataType Structure

Name	Type	Description
ApplyResult	structure	This structure is returned in case of errors. No result data are returned. Further calls with the same TransferId are not possible.
status	Int32	0 – OK – the transferIncidents field may include individual Variables that failed 1 – E_InvalidId – the specified EditContext is unknown
transferIncidents	TransferIncident[]	If the service returns normally and the TransferIncidents list is empty, all changes have been applied and the edited values are cleared. Otherwise, the list contains Variables that could not be transferred successfully. The edited Values are preserved.
contextNodeId	NodeId	The NodeId returned from RegisterNodesById or RegisterNodesByRelativePath.
statusCode	StatusCode	OPC UA StatusCode as defined in IEC 62541-4 and in IEC 62541-8.
diagnostics	DiagnosticInfo	Diagnostic information. This parameter is empty if diagnostics information was not requested in the request header or if no diagnostic information was encountered in the processing of the request. The DiagnosticInfo type is defined in IEC 62541-4.

9.3.7 Reset Method

Clears all modified values in the EditContext.

The signature of this Method is specified below. Table 23 and Table 24 specify the arguments and AddressSpace representation, respectively.

Signature

```

Reset (
    [in] String      EditContextId,
    [out] Int32      ResetStatus);

```

Table 23 – Reset Method Arguments

Argument	Description
EditContextId	Identifier of an EditContext that was previously acquired with a GetEditContext call.
ResetStatus	0 – OK -1 – E_InvalidId – the specified EditContext is unknown

Table 24 – Reset Method AddressSpace Definition

Attribute	Value				
BrowseName	Reset				
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Mandatory

9.3.8 Discard Method

This Method releases the EditContext. Any modified values that have not been applied are lost.

The signature of this Method is specified below. Table 25 and Table 26 specify the arguments and AddressSpace representation, respectively.

Signature

```
Discard(
    [in] String      EditContextId,
    [out] Int32      DiscardStatus);
```

Table 25 – Discard Method Arguments

Argument	Description
EditContextId	Identifier of an EditContext that was previously acquired with a GetEditContext call.
DiscardStatus	0 – OK -1 – E_InvalidId – the specified EditContext is unknown -2 – E_ChildExists – the specified EditContext cannot be discarded, because a child instance exists.

Table 26 – Discard Method AddressSpace Definition

Attribute	Value				
BrowseName	Discard				
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Mandatory

9.4 Direct Device Access

9.4.1 General

DirectDeviceAccess provides the means to communicate with a device in the Device Topology. It will be used by UIs for operations that cannot or at least not easily be performed through Information Model access. Use cases include the transmission of large data buckets from or to the device, for example, historical data or firmware. It is generally assumed that directly accessed data will not be reflected in the Information Model as well. DirectDeviceAccess shall not influence the structure and the data integrity of the Information Model. FDI Servers may be restrictive about when they enable the use of these Methods.

A DirectDeviceAccess Object shall exist for every Device where the FDI Server allows direct access via a UIP.

The following behaviour applies when using DirectDeviceAccess:

- Only one FDI Client can use DirectDeviceAccess at a given time.
- DirectDeviceAccess requires a lock. If the Device has not been locked, the request will be rejected.
- Due to the lock, no write operations and no Method invocations from other FDI Clients are permitted during DirectDeviceAccess. This includes the execution of Actions.
- If Attribute values of Parameters might have changed due to DirectDeviceAccess, the UIP shall set the InvalidateCache in the EndDirectAccess argument to True.

9.4.2 DirectDeviceAccess Type

The DirectDeviceAccessType provides the Methods needed to open and close a connection and to transfer data. Figure 16 shows the DirectDeviceAccessType definition. It is formally defined in Table 27.

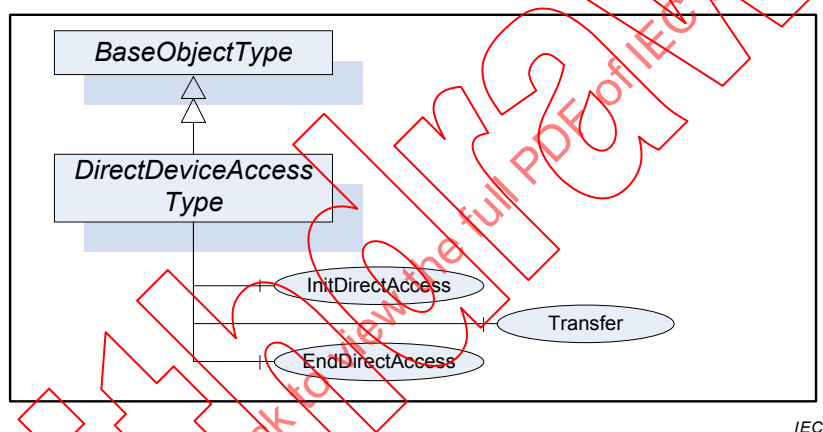


Figure 16 – DirectDeviceAccessType

Table 27 – DirectDeviceAccessType Definition

Attribute	Value				
BrowseName	DirectDeviceAccessType				
IsAbstract	False				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Subtype of the BaseObjectType defined in IEC 62541-5.					
HasComponent	Method	InitDirectAccess			Mandatory
HasComponent	Method	Transfer			Mandatory
HasComponent	Method	EndDirectAccess			Mandatory

The DirectDeviceAccessType and each instance of this Type share the same Methods. The NodeId of these Methods will be fixed and defined in this standard. FDI Clients therefore do not have to browse for these Methods. They can use the fixed NodeId as the MethodId of the Call Service.

The OPC UA StatusCode Bad_MethodInvalid shall be returned from the Call Service for elements where the DirectDeviceAccess Methods are not supported.

9.4.3 DirectDeviceAccess Object

The support of DirectDeviceAccess for an Object is declared by aggregating an instance of the DirectDeviceAccess Type as illustrated in Figure 17.

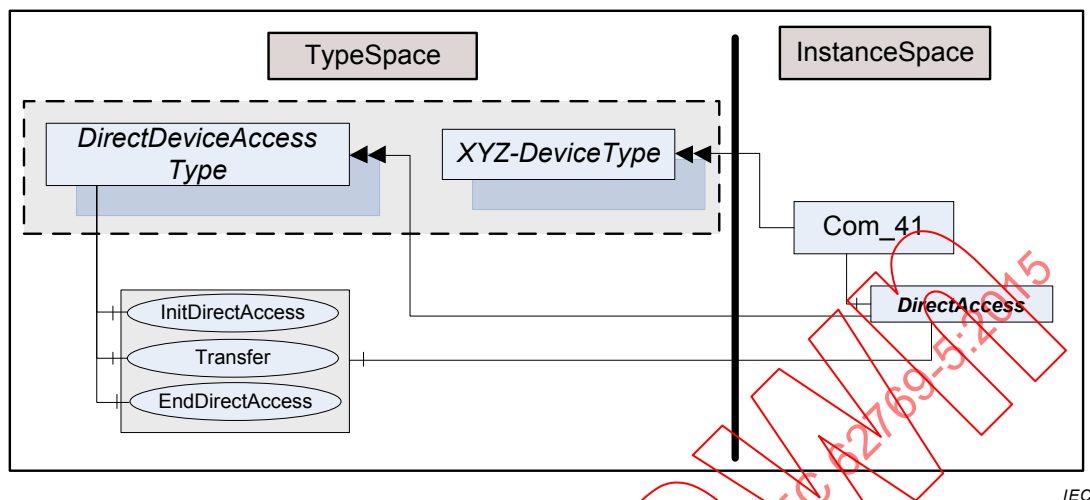


Figure 17 – DirectDeviceAccess instance

This Object is used as container for the DirectDeviceAccess Methods and shall have the BrowseName DirectAccess. It is formally defined in Table 28. HasComponent is used to reference from a TopologyElement (for example a Device) to its “DirectDeviceAccess” Object.

The DirectDeviceAccessType and each DirectAccess Object share the same Methods.

Table 28 – DirectDeviceAccess Instance Definition

Attribute	Value				
BrowseName	DirectAccess				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasTypeDefinition	ObjectType	DirectDeviceAccessType	Defined in 9.4.2.		

9.4.4 InitDirectAccess Method

InitDirectAccess opens a logic communication channel. The Device to access is specified via the ObjectID argument of the Call Service.

It is up to the FDI Server whether this Method already opens a connection to the physical device.

The signature of this Method is specified below. Table 29 and Table 30 specify the arguments and AddressSpace representation, respectively.

Signature

```
InitDirectAccess (
    [in] String      Context,
    [out] Int32      InitDirectAccessError);
```

Table 29 – InitDirectAccess Method Arguments

Argument	Description
Context	A string used to provide context information about the current activity going on in the FDI Client/UIP.
InitDirectAccessError	0 – OK -1 – E_NotSupported – the device can not be directly accessed -2 – E_LockRequired – the element is not locked as required -3 – E_InvalidState – the device is already in DirectAccess mode

Table 30 – InitDirectAccess Method AddressSpace Definition

Attribute	Value				
BrowseName	InitDirectAccess				
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Mandatory

9.4.5 EndDirectAccess Method

EndDirectAccess ends direct access. The directly accessed Device is specified via the ObjectId argument of the Call Service.

The signature of this Method is specified below. Table 31 and Table 32 specify the arguments and AddressSpace representation, respectively.

Signature

```
EndDirectAccess (
    [in] Boolean InvalidateCache,
    [out] Int32 EndDirectAccessError);
```

Table 31 – EndDirectAccess Method Arguments

Argument	Description
InvalidateCache	If True, the FDI Server will invalidate any cached values for Device Parameters. This means that these Parameters will be re-read before they are used again.
EndDirectAccessError	0 – OK -1 – E_InvalidState – the device is not in DirectAccess mode.

Table 32 – EndDirectAccess Method AddressSpace Definition

Attribute	Value				
BrowseName	EndDirectAccess				
References	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Mandatory

9.4.6 Transfer Method

Transfer is used to transfer data to and from the Device. The format of send or receive data is protocol specific.

The signature of this Method is specified below. Table 33 and Table 34 specify the arguments and AddressSpace representation, respectively.

Signature

```
Transfer (
    [in] String      SendData,
    [out] String     ReceiveData,
    [out] Int32      TransferError);
```

Table 33 – Transfer Method Arguments

Argument	Description
SendData	XML document based on the TransferSendDataType as specified in the communication profile-specific XML schema. See IEC 62769-4:2015, Annex F and IEC 62769-10*.*.
ReceiveData	XML document based on the TransferResultDataType as specified in the communication profile-specific XML schema. See IEC 62769-4:2015, Annex F and IEC 62769-10*.*.
TransferError	0 – OK -1 – E_InvalidState – the device is not in DirectAccess mode.

Table 34 – Transfer Method AddressSpace Definition

Attribute	Value				
BrowseName	Transfer				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Mandatory

10 Parameter Types

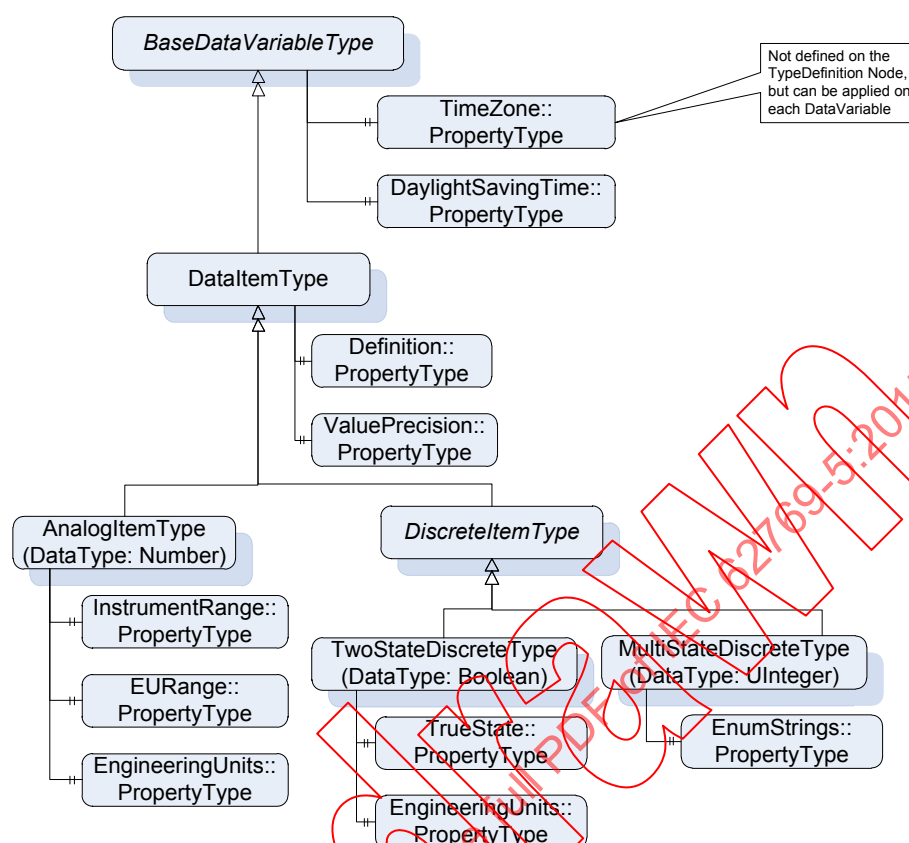
10.1 General

IEC 62541-100 defines a Parameter as “a variable of the Device that can be used for configuration, monitoring or control purposes. In the Information Model it is synonymous to an OPC UA DataVariable.”

When discussing Parameter Types we have to consider DataType and VariableType.

Each OPC UA DataVariable (Parameter) has a Value Attribute, which is of a certain DataType. OPC UA has already defined a set of built-in DataTypes, which are sufficient for the majority of types needed for FDI.

VariableTypes represent the type definition of (Data)Variables. Such a type definition typically defines certain behaviour as well as mandatory or optional Properties. The HasTypeDefinition Reference is used to define the VariableType for a Variable. OPC UA already defines a set of VariableTypes as illustrated in Figure 18. BaseDataVariableType is the base type and can be used if no more specialized type information is available (see IEC 62541-5). AnalogItemType and DiscreteItemType (see IEC 62541-8) are more concrete types.



IEC

Figure 18 – OPC UA VariableTypes including OPC UA Data Access

The mapping of EDD Data Types to OPC UA DataTypes and VariableTypes is specified in 15.5.

10.2 ScalingFactor Property

The Value Attribute of Variables contains the raw value returned from the device. However, Servers can expose the Property ScalingFactor. It is suggested, that the (raw) value is multiplied by this factor before being displayed.

The Property shall be aggregated by each Variable that it applies to. It is formally defined in Table 35.

Table 35 – ScalingFactor Property Definition

Name	DataType	Description
ScalingFactor	Double	This Property specifies the scaling factor to be used when displaying the Variable value.

10.3 Min_Max_Values Property

This Property specifies one or more ranges to which a Variable value shall be set. If there are multiple ranges they shall not overlap. A “Null” for the MIN_Value or the MAX_Value indicates that this boundary is not limited.

The Property shall be aggregated by each Variable that it applies to. It is formally defined in Table 35.

Table 36 – Min_Max_Values Property Definition

Name	Data Type	Description
Min_Max_Values	Variant_Range[]	This Property specifies the range or ranges to which a Variable Value shall be set.

The Variant_Range structure specifies a single range (a MIN_Value and a MAX_Value). The BaseDataType is used, because such a range can be applied to Variables of different DataTypes – in particular integer, floating point, date and duration. The actual datatype used has to match the DataType of the Variable value. It can also be Null which means that this boundary is not defined.

Its elements are defined in Table 37.

Table 37 – Variant_Range DataType Structure

Name	Type	Description
Variant_Range	structure	
MIN_Value	BaseDataType	Specifies the upper bound of values to which a Variable shall be set. Null indicates that MIN_Value has no limit.
MAX_Value	BaseDataType	Specifies the lower bound of values to which a Variable shall be set. Null indicates that MAX_Value has no limit.

Its representation in the AddressSpace is defined in Table 38.

Table 38 – Variant_Range Definition

Attributes	Value
BrowseName	Variant_Range

11 FDI StatusCodes

Clause 11 defines OPC UA StatusCodes that are specific to FDI Servers.

The general structure of the StatusCode is specified in IEC 62541-4. It includes a set of common operational result codes which also apply to FDI.

Table 39 contains Good (success) codes that are specifically defined for FDI.

Table 39 – Good operation level result codes

Symbolic Id	Description
Good_Edited	This status applies to Variable values that are part of an EditContext (see 9.3). It is returned with values that are read or received in a DataChangeNotification from a Subscription. It defines that it is an edited value that has not been transferred from the EditContext to the Device.
Good_PostActionFailed	The value of a Variable was successfully read or written but one of the post actions failed.
Good_DependentValueChanged	This status applies to Variable values that are part of an EditContext (see 9.3). It is delivered with a dominant Variable value when any of its dependent Variables was changed and not yet applied.

12 Specialized topology elements

Devices and other topology elements can be specialized using the modelling elements defined in this standard or in IEC 62541-100. No specific ObjectType is needed. A

Communication Device for instance will have the CommunicationServices capability (Communication Devices and CommunicationServices are specified in IEC 62769-7).

Other specializations are possible applying the same techniques, for example, a Modular Communication Device.

See IEC 62541-100 for the definition of Block Device and Modular Device.

13 Auditing

13.1 General

Auditing is a requirement in many systems. It provides a means for tracking activities that occur as part of the normal operation of the system. It also provides a means for tracking abnormal behaviour. It is also a requirement from a security standpoint.

When an audit trail is maintained by the FDI Server, all audit trail records related to services invoked by FDI Clients will be implicitly created. In addition, FDI Clients have means for providing additional audit context information as specified in 13.2 and 13.3.

FDI Servers shall generate AuditEvents as specified in IEC 62541-4 and IEC 62541-5. These standards define AuditEvents for the following OPC UA Services:

- Secure Channel Services
- Session Services
- NodeManagement Service Set (AddNode, DeleteNode)
- Write Service
- Method Service ("Call")

No AuditEvents are defined for reading and for subscriptions.

The FDI Server generates AuditEvents using the standard OPC UA event mechanism. This allows an external OPC UA Client to subscribe to and store the audit entries in a log file or other storage location.

13.2 FDI Client-provided context information

FDI Clients have various means for providing context information for the purpose of auditing:

- the ClientDescription and SessionName passed by the FDI Client when creating the Session,
- the context text from the Methods to initialise Lock or Direct Device Access.

When an AuditUpdateMethodEvent is generated for the EnterLock and InitDirectAccess Methods, the audit context information will be used for the Message field of this event.

13.3 LogAuditTrailMessage Method

LogAuditTrailMessage is used by the FDI Client to insert information about the current activity going on in the FDI Client into the audit trail of the FDI Server. Since it is a Method, it will be also inserted into the stream of AuditEvents. The message will be timestamped in the FDI Server.

This Method shall be a component of the Server Object. The ObjectId parameter of the Call Service shall be the NodId of the Server Object.

FDI Clients do not have to browse for the LogAuditTrailMessage Method. Rather they can use the well known NodeId of the Method declaration as the MethodId of the Call Service.

The signature of this Method is specified below. Table 40 and Table 41 specify the arguments and AddressSpace representation, respectively.

Signature

```
LogAuditTrailMessage (
    [in] String      Message);
```

Table 40 – LogAuditTrailMessage Method Arguments

Argument	Description
Message	Free text describing the context.

Table 41 – LogAuditTrailMessage Method AddressSpace Definition

Attribute	Value				
BrowseName	LogAuditTrailMessage				
References	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Mandatory

When an AuditUpdateMethodEvent is generated for the LogAuditTrailMessage Method, the Message argument will be used for the Message field of this event.

14 FDI Server Version

The FDI Technology Version supported by an FDI Server is exposed via an FDI-specific Property. The version exposed with this Property applies to

- the Information Model, and
- the XML Schema as used for UIDs and Actions.

The Property shall be aggregated by the OPC UA Server Object. It is formally defined in Table 42.

Table 42 – FDI Server Version Property Definition

Name	DataType	Description
FDIServerVersion	String	This Property specifies the FDI Technology Version that this FDI Server supports. The syntax of the string is as defined in IEC 62769-4.

15 Mapping FDI Package information to the FDI Information Model

15.1 General

Clause 15 defines the mapping of EDDL and other FDI Package Information to the FDI Information Model and the underlying OPC UA and OPC UA Devices information models, respectively.

The OPC UA Object Model provides a standard way for Servers to represent Objects to Clients. In order to meet this objective, the OPC UA Object Model allows the definition of Objects in terms of Variables and Methods. It also allows relationships to other Objects to be expressed.

On the other hand, EDDL defines a set of language constructs that are used to describe industrial field devices. Each construct supports its own set of attributes and references. EDD information adds semantic contents to the raw data values read from and written to the field devices.

The primary objective of the EDDL-OPC UA information model is to describe the correspondence between the OPC UA Object Model elements and the EDDL elements when an EDD is used to populate the FDI Server with Objects.

Completely meeting this objective means dealing both with enhanced data access through OPC UA and with UI interactions between OPC UA clients and servers.

15.2 Localization

15.2.1 Localized text

In various definitions, EDDs may contain information in multiple language variants. Examples are the LABEL and HELP attributes. The server has to select the proper language for each client as follows: When creating an OPC UA Session, the OPC UA Client passes the locale(s) that it requests to be used for all services within this Session. If the Server does not support any of the requested locales it chooses an appropriate default locale.

15.2.2 Engineering units

Variables with a UNIT CODE are represented in OPC UA as AnalogItem Variables. The UNIT CODE is exposed via the EngineeringUnit Property. Changing the EngineeringUnit will cause all EDD Variables that depend on the associated UNIT CODE to be recalculated. As a result the OPC UA Variable values will be set as well.

Changing the EngineeringUnit will affect all Clients.

15.3 Device

15.3.1 General

Subclause 15.3 specifies the mapping of FDI Package elements to Device Types and Device Instances.

Devices may also have Blocks (see 15.4).

15.3.2 Mapping to Attributes to a specific DeviceType Node

The BrowseName and NodeId Attributes are vendor specific.

The DisplayName Attribute is created from the Package Catalog: DeviceType.Name.

The Description Attribute of a Device is information that serves to further identify, manage, locate, and/or explain the device whose contents are defined by the user. For purely block-oriented devices this will appear only on blocks. For example, in HART the MESSAGE variable can be used here.

15.3.3 Mapping to Properties

Type and instances share the same Properties.

Some of the Properties are created from information in the Package Catalog. Table 43 specifies the mapping of Properties to Package Catalog elements.

Table 43 – DeviceType Property Mapping

Property	Package Catalog element
Manufacturer	ManufacturerName
Model	ListOfDeviceTypes[i].ListOfInterfaces[j].DeviceModel.
DeviceRevision	ListOfDeviceTypes[i].ListOfInterfaces[j].Version.
FDITechnologyVersion	FDITechnologyVersion.

The SerialNumber, RevisionCounter, SoftwareRevision, and HardwareRevision Properties correspond to the value of the protocol-specific “serial number”, “revision counter”, “software revision”, and “hardware revision” Parameters, respectively.

The DeviceManual Property is an empty string. Documents can be found in the attachment set.

If a Picture element is available in the FDI Package it can be mapped to the OPC UA Icon Property (see IEC 62541-3). If multiple resolutions are available the host has to choose.

15.3.4 Mapping to ParameterSet

Some devices are strictly block-oriented with no kinds of Variables on the device-level. Mapping an EDD for these devices will result in an empty ParameterSet. When VARIABLE, VALUE_ARRAY or LIST items exist on the device-level, the resulting Parameters are kept in the “ParameterSet” as a flat list of Parameters. FunctionalGroups reflect the structure of the device menus defined in the device’s EDD.

A ParameterSet with all Parameters exists on the Type, the offline and the online instances.

See 15.5 on how EDDL attributes are used to create Parameter nodes.

15.3.5 Mapping to Functional Groups

The top-level Functional Groups that are referenced directly from the Device Object correspond to the root MENU items as defined in IEC 61804-4. Naming conventions are used to differentiate between Functional Groups for handheld and for PC-based applications.

There are no Functional Groups on the DeviceType.

15.3.6 Mapping to DeviceTypeImage

The Variables in the DeviceTypeImage folder are created from the Package Catalog: DeviceTypes[i].ListOfDeviceImages. BrowseName and DisplayName represent the file name from the Relationship Type.

15.3.7 Mapping to Documentation

The Variables in the Documentation folder are created from the Package Catalog: DeviceTypes[i].ListOfDocuments. BrowseName and DisplayName represent the file name from the Relationship Type.

15.3.8 Mapping to ProtocolSupport

The Variables in the ProtocolSupport folder are created from the Package Catalog: DeviceTypes[i].ListOfInterfaces[j].ListOfCommunicationProfileSupportFiles. BrowseName and DisplayName represent the file name from the Relationship Type.

15.3.9 Mapping to ImageSet

The ImageSet contains Variable Nodes for all images from the EDD that are needed for UIDs.

15.3.10 Mapping to ActionSet

The ActionSet references OPC UA Objects that represent all EDD Methods except for the abort and the action methods as defined in IEC 61804-3. See 15.7 on how EDDL attributes are used to create Action Nodes

15.3.11 Mapping to MethodSet

The MethodSet (inherited from IEC 62541-100) is not used by the FDI.

15.4 Block

15.4.1 General

Subclause 15.4 specifies the mapping of EDDL elements to Block Types and instances.

EDDL supports the definition of devices that are block-oriented, and devices that are non-block oriented. When blocks (specifically, Block_A) exist in an EDD, the resulting device shall be modelled as block-oriented Device as specified in IEC 62541-100. The Block node instances are kept in the Blocks component. A Device that does not support EDDL defined Blocks will not have the Blocks component.

The SupportedTypes folder in the Blocks component of a DeviceType references all BlockTypes that may appear in a Device Instance. The Blocks component in a Device Instance references instantiated Blocks.

15.4.2 Mapping to Attributes

The BrowseName and DisplayName correspond to the EDD Identifier and LABEL Attribute of the corresponding EDDL Block, respectively.

The NodeId is vendor specific.

The Help attribute of the corresponding EDDL Block is mapped to the Description attribute of the Block instance node. Bad_AttributeInvalid shall be used if EDD contains no Help.

15.4.3 Mapping to ParameterSet

All VARIABLE, VALUE_ARRAY, LIST items specified for a Block are used as Parameters in the "ParameterSet". A ParameterSet with all Parameters exist on the Type, the offline and the online instances.

See 15.5 on how EDDL attributes are used to create Parameter nodes.

15.4.4 Mapping to Functional Groups

A Block may have FunctionalGroups that expose its Parameters in an organized fashion, reflecting the structure of the block menus defined in the device's EDD. The top-level Functional Groups that are referenced directly from the Block Object correspond to the root MENU items as defined in IEC 61804-4. Naming conventions are used to differentiate between Functional Groups for handheld and for PC-based applications

The BrowseName of a FunctionalGroup is the EDD identifier of the corresponding EDDL MENU or COLLECTION.

There are no Functional Groups on the BlockType.

15.4.5 Mapping to ActionSet

The ActionSet references OPC UA Objects that represent all EDD Methods defined for a specific BLOCK except for the abort and the action methods as defined in IEC 61804-3. See 15.7 on how EDDL attributes are used to create Action Nodes

15.4.6 Mapping to MethodSet

The MethodSet (inherited from IEC 62541-100) is not used by the FDI.

15.4.7 Instantiation rules

The BrowseName of a Block is generated by the FDI Server from the EDD identifier of the corresponding EDDL BLOCK_A by adding a numeric suffix that the OPC UA server generates in order to make the BrowseName unique. For example, __analog_input_0, __analog_input_1, __pid_control_0.

The DisplayName of a BLOCK_B Block is the LABEL attribute.

The DisplayName of a BLOCK_A Block is defined in the protocol annex.

The Description of a Block is the HELP attribute of the corresponding EDDL BLOCK. Bad_AttributeIdInvalid shall be used if the EDD contains no Help.

15.5 Parameter

15.5.1 General

EDDL Parameters (for devices or blocks) are mapped to OPC UA Variables. The VariableType used may be of any sub-type of the abstract BaseVariableType. In most cases they will be mapped to the VariableTypes defined in IEC 62541-8, for example, DataItemtype, AnalogItemtype or DiscreteItemtype. This standard includes additional VariableTypes for more sophisticated EDDL types. See Table 45 for the mapping of EDDL types.

The BrowseName of a Parameter is the EDD identifier of the corresponding EDDL VARIABLE, RECORD or VALUE_ARRAY.

The DisplayName of a Parameter is the LABEL attribute of the corresponding EDDL VARIABLE, RECORD or VALUE_ARRAY.

The Description of a Parameter is the HELP attribute of the corresponding EDDL VARIABLE, RECORD or VALUE_ARRAY. Bad_AttributeIdInvalid shall be used if the EDD contains no Help.

Parameters have also a set of Attributes that are common to all VariableTypes. Table 44 summarizes the Variable Attributes and describes how they are set from the EDDL description information.

Table 44 – Setting OPC UA Variable Attributes from EDDL variable attributes

Attributes	Description									
Value	The most recent value of the Variable that the FDI Server has read from the device.									
DataType	The EDDL data type is translated into an OPC UA standard data type according to Table 45.									
ValueRank	Either set to “Scalar” or – when the Parameter is an array – the NUMBER_OF_ELEMENTS specified for the EDDL VALUE_ARRAY item.									
AccessLevel	The AccessLevel Attribute is set based on the EDDL variable HANDLING attribute according to the following table: <table><tr><th>Field</th><th>Bit</th><th>Description</th></tr><tr><td>CurrentRead</td><td>0</td><td>Set if EDDL variable HANDLING is defined as READ. Reset otherwise.</td></tr><tr><td>CurrentWrite</td><td>1</td><td>Set if EDDL variable HANDLING is defined as WRITE. Reset otherwise.</td></tr></table> If the HANDLING attribute is missing, the Parameter will be defined as readable and writeable.	Field	Bit	Description	CurrentRead	0	Set if EDDL variable HANDLING is defined as READ. Reset otherwise.	CurrentWrite	1	Set if EDDL variable HANDLING is defined as WRITE. Reset otherwise.
Field	Bit	Description								
CurrentRead	0	Set if EDDL variable HANDLING is defined as READ. Reset otherwise.								
CurrentWrite	1	Set if EDDL variable HANDLING is defined as WRITE. Reset otherwise.								
UserAccessLevel	The AccessLevel with possible restrictions based on client identity as defined by the FDI Server.									
MinimumSamplingInterval	The MinimumSamplingInterval Attribute indicates how fast the FDI Server can reasonably sample the value for changes. It is suggested that the FDI Server checks the variable CLASS attribute to differentiate static variables from dynamic variables regarding the sampling interval. Static variables might be sampled only once and then only when the RevisionCounter changes. For static variables the FDI Server can use the MinimumSamplingInterval -1 (indeterminate).									

The Value Attribute is the Parameter value, and – for the online representation – reflects the device data value.

The DataType Attribute is an OPC UA DataType chosen to match the EDDL type and size. Table 45 shows the correspondence between the EDDL types and sizes and the OPC UA VariableTypes and standard DataTypes.

Table 45 – Correspondence between EDDL and OPC UA standard data types

EDDL Data Type	OPC UA VariableType	OPC UA DataType	Constraints
Arithmetic			
INTEGER	BaseDataVariableType, AnalogItemType	SByte	When the size specified in EDDL is 1 byte.
		Int16	When the size specified in EDDL is 2 bytes.
		Int32	When the size specified in EDDL is 3 or 4 bytes.
		Int64	When the size specified in EDDL is 5, 6, 7 or 8 bytes.
UNSIGNED_INTEGER	BaseDataVariableType, AnalogItemType	Byte	When the size specified in EDDL is 1 byte.
		UInt16	When the size specified in EDDL is 2 bytes.
		UInt32	When the size specified in EDDL is 3 or 4 bytes.
		UInt64	When the size specified in EDDL is 5, 6, 7 or 8 bytes.
DOUBLE	BaseDataVariableType, AnalogItemType	Double	
FLOAT	BaseDataVariableType, AnalogItemType	Float	
ENUMERATED	See 15.5.5	Byte	When the size specified in EDDL is 1 byte.
		UInt16	When the size specified in EDDL is 2 bytes.
		UInt32	When the size specified in EDDL is 3 or 4 bytes.
		UInt64	When the size specified in EDDL is 5, 6, 7 or 8 bytes.
BIT_ENUMERATED	See 15.5.6	Byte	When the size specified in EDDL is 1 byte.
		UInt16	When the size specified in EDDL is 2 bytes.
		UInt32	When the size specified in EDDL is 3 or 4 bytes.
		UInt64	When the size specified in EDDL is 5, 6, 7 or 8 bytes.
Date-and-Time			
DATE	BaseDataVariableType	UtcTime (a 64-bit signed integer which represents the number of 100 nanosecond	The data type DATE consists of a calendar date. This data type has special codification in the device. Conversion to UtcTime is necessary when reading it from the device. Conversion back to DATE is necessary when writing it to the device. On write, if invalid data is written, the

EDDL Data Type	OPC UA VariableType	OPC UA DataType	Constraints
		intervals since January 1, 1601 (UTC))	service returns an appropriate error in the serviceResult and the diagnosticInfo data members of the ResponseHeader and does not accept the data.
DATE_AND_TIME	BaseDataVariableType	UtcTime	The data type DATE_AND_TIME consists of a calendar date and a time. This data type has special codification in the device. Conversion to UtcTime is necessary when reading it from the device. Conversion back to DATE_AND_TIME is necessary when writing it to the device. On writing if invalid data is written the service returns an appropriate error in the serviceResult and the diagnosticInfo data members of the ResponseHeader and does not accept the data.
DURATION	BaseDataVariableType	Duration (a Double that defines an interval of time in milliseconds (fractions can be used to define sub-millisecond values))	The DURATION data type is a time difference that consists of a time in milliseconds and an optional day count. This data type has special codification in the device. Conversion to Time is necessary when reading it from the device. Conversion back to DURATION is necessary when writing it to the device. On writing, if invalid data is written, the service returns an appropriate error in the serviceResult and the diagnosticInfo data members of the ResponseHeader and does not accept the data. The OPC Duration type is limited to approximately 49,7 days, which is less than the theoretical maximum of the EDDL DURATION type. If the DURATION exceeds the OPC maximum the quality should be set to indicate this condition.
TIME	BaseDataVariableType	UtcTime	The TIME data type consists of a time and an optional date. This data type has special codification in the device. Conversion to UtcTime is necessary when reading it from the device. Conversion back to TIME is necessary when writing it to the device. On writing, if invalid data is written, the service returns an appropriate error in the serviceResult and the diagnosticInfo data members of the ResponseHeader and does not accept the data.
TIME_VALUE[4]	BaseDataVariableType	Duration	Number of 1/32 ms.
TIME_VALUE[8]	BaseDataVariableType	UtcTime	The data type TIME_VALUE is used to represent date and time in the required precision for application clock synchronization. This data type has special codification in the device. Conversion to UtcTime is necessary when reading it from the device. Conversion back to TIME_VALUE is necessary when writing it to the device. On writing, if invalid data is written, the service returns an appropriate error in the serviceResult and the diagnosticInfo data members of the ResponseHeader and does not accept the data.
String			
ASCII	BaseDataVariableType	String	Conversion to String is necessary when reading it from the device. Conversion back to ASCII is necessary when writing it to the device. On writing, if invalid characters are written, the service returns an appropriate error in the serviceResult and the diagnosticInfo data members of the ResponseHeader and does not accept the string.
BIT_STRING	BaseDataVariableType	ByteString	Conversion to ByteString is necessary when reading it from the device. Conversion back to BIT_STRING is necessary when writing it to the device. On writing, if invalid characters are written, the service returns an appropriate error in the serviceResult and the diagnosticInfo data members of the ResponseHeader and does not accept the string.
EUC	BaseDataVariableType	String	Conversion to String is necessary when reading it from the device. Conversion back to EUC is necessary when writing it to the device. On writing, if invalid characters are written, the

EDDL Data Type	OPC UA VariableType	OPC UA DataType	Constraints
			service returns an appropriate error in the serviceResult and the diagnosticInfo data members of the ResponseHeader and does not accept the string.
PACKED_ASCII	BaseDataVariableType	String	Conversion to String is necessary when reading it from the device. Conversion back to PACKED_ASCII is necessary when writing it to the device. On writing, if invalid characters are written, the service returns an appropriate error in the serviceResult and the diagnosticInfo data members of the ResponseHeader and does not accept the string.
PASSWORD	BaseDataVariableType	String	Conversion to String is necessary when reading it from the device. FDI Servers shall allow PASSWORD Variables to be read only when a secure OPC UA channel has been created, i.e., a channel with encryption.
VISIBLE	BaseDataVariableType	String	Conversion to String is necessary when reading it from the device. Conversion back to VISIBLE is necessary when writing it to the device. On writing, if invalid characters are written, the service returns an appropriate error in the serviceResult and the diagnosticInfo data members of the ResponseHeader and does not accept the string.
OCTET	BaseDataVariableType	ByteString	
INDEX	see 15.5.5	String	
BOOLEAN	BaseDataVariableType	Boolean	

As Table 45 shows, EDDL supports a variety of variable types. While that table shows the correspondence between the EDDL data types and the OPC UA VariableType, it does not provide any details on how other EDDL VARIABLE TYPE construct attributes are supported. It does not provide any details on how other OPC UA BaseDataVariableType attributes should be set either. Subclause 15.5 is intended to provide details for each EDDL data type.

15.5.2 Private Parameters

The Parameters specified in an FDI Package may be declared private using the PRIVATE Attribute specified in 61804-3. The FDI Server shall create Nodes in the Information Model for the private Parameters but they shall not be browsable. The FDI Server shall return the NodeIds of private Parameters when the name of such a Parameter is passed to TranslateBrowsePathsToNodeIds (the startingNode argument shall be the "ParameterSet" Object). Once the FDI Client has obtained the NodeId all Service requests for private Parameters will be processed in the same way as for public (browsable) Parameters.

An example of private parameters is parameters that should only be modified through an Action. These parameters should not be visible to FDI Clients to prevent direct access. FDI Clients invoke Actions to access these private parameters.

15.5.3 MIN_Value and MAX_Value

If one or more MIN_VALUE and MAX_VALUE attributes are specified for a variable in EDDL they shall be mapped to the Min_Max_Values Properties defined in 10.3.

15.5.4 Engineering units

The EngineeringUnits Property defined in IEC 62541-3 and IEC 62541-8 shall be used:

- If the variable has the CONSTANT_UNIT defined for itself in EDDL. In this case the EngineeringUnits Property keeps the EDD CONSTANT_UNIT variable attribute. The FDI Server shall deal with the fact that CONSTANT_UNIT can be conditional.

- If the variable is involved in an EDD UNIT relation it is the FDI Server's responsibility to implement the unit code update mechanism. An EDD UNIT relation specifies a reference to a variable holding a unit code and a list of dependent variables. When the variable holding the unit code is modified, the list of effected variables using that unit code shall be refreshed. When an effected variable is displayed, the value of its unit code shall also be displayed.

The standard OPC UA notifications can be used to report to the FDI Clients that the unit code changed.

15.5.5 Enumerated Parameters

An OPC UA DataVariable with the OPC UA DataType Enumeration is used for each enumerator in the EDDL enumerated variable definition.

The Value Attribute of the DataVariable is the numeric value of the state, and corresponds to the value attribute of the EDDL ENUMERATED TYPE.

The ValueAsText Property of the DataVariable exposes the display value of the state, which corresponds to the description attribute of the EDDL ENUMERATED TYPE.

The EnumValues Property of the DataVariable contains the complete list of enumerations, i.e., a table where each element is a structure consisting of EDDL ENUMERATED TYPE attributes "value", "description", and "help". If no help attribute exists in the EDD, the value of the description attribute will also be used for this purpose.

15.5.6 Bit-enumerated Parameters

A DataVariable of OPC UA OptionSet VariableType is used for each EDDL BIT ENUMERATED VARIABLE definition. The OPC UA DataType is an array of Boolean where one Boolean is used per bit in the EDDL BIT_ENUMERATED VARIABLE definition.

The Boolean array of the DataVariable reflects the status of all bits. TRUE is used when a bit is set and FALSE when a bit is reset.

The EnumValues Property of the DataVariable contains the complete list of bit enumerations, i.e., a table where each element is a structure consisting of EDDL BIT ENUMERATED TYPE attributes "bit position", "description", and "help". If no help attribute exists in the EDD, the value of the description attribute will also be used for this purpose.

15.5.7 Representation of records

A complex DataVariable is used to represent EDDL RECORD Parameters. The root DataVariable represents the record itself. It will have component DataVariables that represent the EDDL RECORD MEMBERS. (The MEMBERS of an EDDL RECORD are defined in EDDL by means of a reference to an EDDL VARIABLE.).

See Figure 19 for an example of how records are represented in the OPC UA AddressSpace.

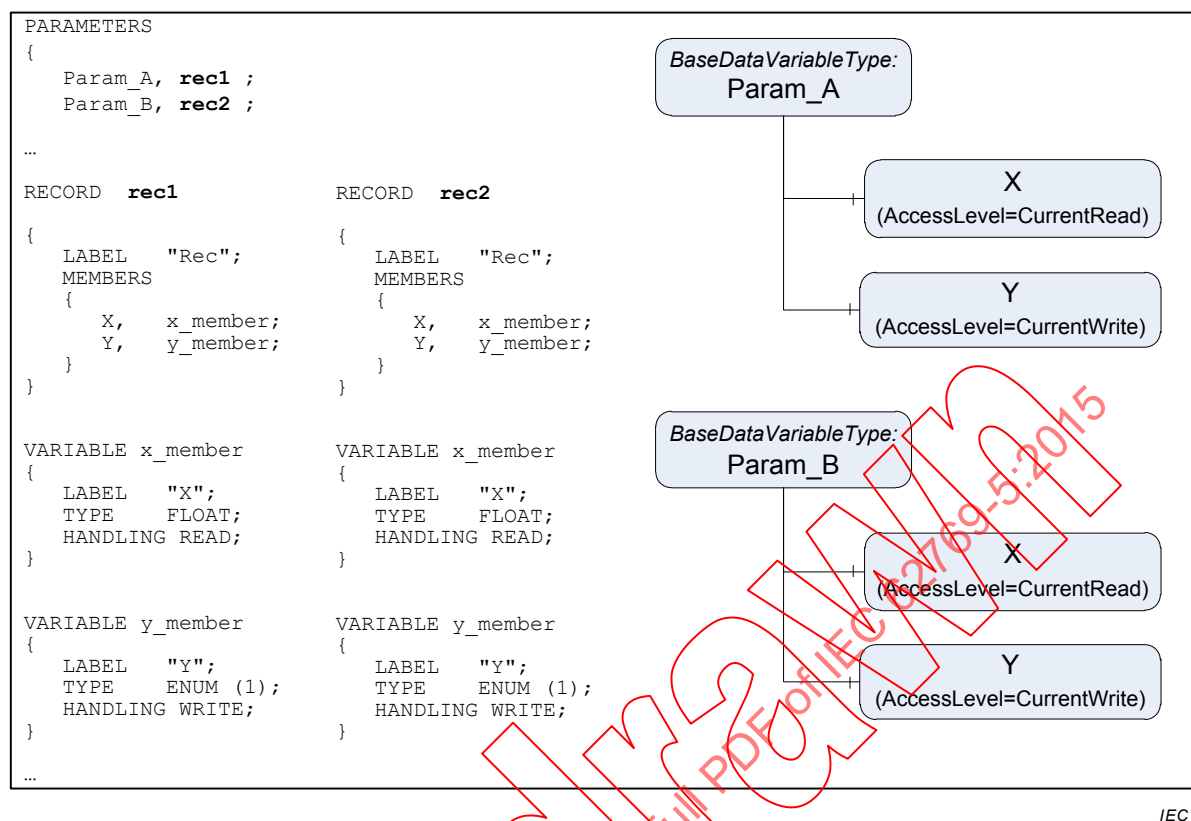


Figure 19 – Example: Complex variable representing a RECORD

BrowseName and DisplayName of the root DataVariable are set to the EDD identifier of the EDDL VARIABLE that implements this RECORD type. The DataType Attribute of the “root” DataVariable is BaseDataType. The ValueRank Attribute is used to specify that the value contains an array. The Value Attribute represents the values of all members in the order as defined for the RECORD. According to the example in Figure 19, the first variant will contain a floating point value and the second will be a numeric value representing the Enumeration.

For each component DataVariable that represents an EDDL RECORD MEMBER:

- The BrowseName is the identifier of the corresponding EDDL VARIABLE.
- The DisplayName is the LABEL attribute of the corresponding EDDL VARIABLE.
- The Description is the HELP attribute of the corresponding EDDL VARIABLE. Bad_AttributeInvalid shall be used if the EDD contains no Help.
- The AccessLevel is derived from the HANDLING attribute. Readable and Writeable shall be used if the EDD contains no HANDLING attribute.

15.5.8 Representation of arrays, and lists of Parameters with simple data types

A single DataVariable will represent an EDDL VALUE_ARRAY or LIST item when the data type of the referenced array element has a simple data type.

The OPC UA DataVariable Attributes are set as follows:

- DataType is set to the type of the array element (see Table 45 for the data type mapping).
- The ValueRank Attribute is used to specify that the value contains an array. In case of an EDDL VALUE_ARRAY the number of elements is exposed. In the case of an EDDL LIST the number of elements is unspecified since the size can change dynamically.

15.5.9 Representation of values arrays, and lists of RECORD Parameters

Value arrays or lists of non-simple data types will be represented as OPC UA array of complex variables. Figure 20 shows the EDDL sample code of a VALUE_ARRAY of RECORDs and the corresponding complex variable in the OPC UA AddressSpace.

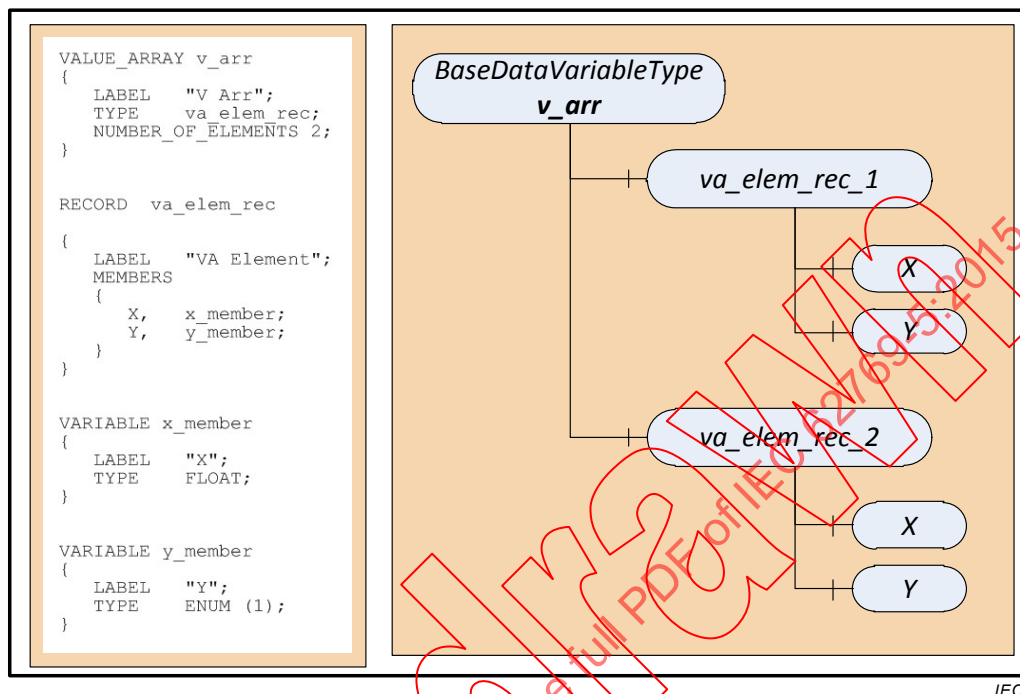


Figure 20 – Complex variable representing a VALUE_ARRAY of RECORDs

In the FDI Server AddressSpace a complex DataVariable is used to represent the entire VALUE_ARRAY. Each VALUE_ARRAY element, which is in fact a RECORD, is represented as a component complex DataVariable. The RECORD MEMBERS are also represented as component DataVariables. The FDI Client refers to the X member of the first record in the array through the BrowseName v_arr.va_elem_1.x_member. Note that the index always begins with '1'.

The DataType Attribute of all complex root DataVariables is BaseDataType. The ValueRank Attribute is used to specify that the value contains an array. The Value Attribute represents all VALUE_ARRAY entries. The first element corresponds to the first array entry and so on. Each element in turn contains an array. This may either be an array of simple types or an array of BaseDataType. A RECORD is always represented as an array of BaseDataType.

15.5.10 Representation of COLLECTION and REFERENCE ARRAY

The EDDL constructs COLLECTION and REFERENCE ARRAY are mapped to Functional Group Objects (see 15.6) when used for grouping only.

15.5.11 SCALING_FACTOR

If the EDD includes a SCALING_FACTOR attribute for an EDD VARIABLE it shall be mapped as follows.

- The Parameter value and associated Properties (like Ranges) shall be exposed in raw format (not scaled by the FDI Server). Any scaling is responsibility of the FDI Client (or UIP, respectively).

- The SCALING_FACTOR shall be provided in a Property with the BrowseName “ScalingFactor”. The DataType shall be “Double”. If the SCALING_FACTOR in the EDD is an expression, this expression will be computed by the EDD Interpreter.
- The SCALING_FACTOR is also included in UID documents.

15.6 Functional Groups

FunctionalGroups are used to group Variables (EDDL parameters) or Actions (EDDL methods). They have a recursive definition, that is, FunctionalGroups may reference other FunctionalGroups.

Both Devices and Blocks may have FunctionalGroups. EDDL MENUs, EDDL BLOCK_A PARAMETERS, COLLECTIONS, and REFERENCE ARRAYS are the building blocks for FunctionalGroups. The proper attributes of these EDDL constructs control which elements are referenced by the Functional Group.

- VARIABLE, RECORD, or VALUE_ARRAY will cause a reference to the corresponding FDI Parameter.
- METHODS will be used to organise FDI Action Objects
- Elements that represent one of the building blocks again will cause a reference to a subordinate Functional Group.

Root MENUs as defined in IEC 61804-4 will be top-level Functional Groups. The FDI Server generates the whole hierarchy of FunctionalGroups by browsing the EDDL MENU and their ITEMS attribute definitions. The FunctionalGroups reproduce the same structure of the MENUs as they are defined in EDD for the Device or Block.

All FunctionalGroups are instantiated in both the offline and the online representation of a Device. Naming conventions defined in IEC 61804-4 can be used to determine if a FunctionalGroup is relevant for offline or online.

The BrowseName Attribute of a FunctionalGroup is the EDD identifier of the menu or the instance identifier of the block. The naming conventions defined in IEC 61804-4 can be used to determine if a FunctionalGroup is relevant for PC or handheld.

The DisplayName Attribute of a FunctionalGroup is the LABEL attribute of the corresponding building block.

The Description Attribute of a FunctionalGroup is the value of the HELP attribute of the corresponding building block. Bad_AttributeInvalid shall be used if the EDD contains no Help.

FunctionalGroups exposing standard sets of Parameters such as diagnostic information will be created for an EDD, which comply with the menu conventions for PC-based application as defined in IEC 61804-4.

15.7 AXIS elements in UIDs

AXIS elements are not represented as instances in the FDI information model. However, an EDDL AXIS has writable VIEW_MIN/VIEW_MAX attributes. They shall be modified by the FDI Client to inform the server about the zooming, scrolling or positioning.

To enable this modification, the FDI Server shall create variable nodes for each of these attributes in the Information Model. They need not be browsable. The NodeIds of these nodes are included into the UID document as described in the UID schema.

15.8 Actions

FDI Actions are used to represent EDD METHODS. Only EDD METHODS that are to be called by the FDI Client as part of a UID or a UIP shall be exposed as FDI Actions. If METHODS defined in IEC 61804-3, which include the pre or post read, edit and write actions are exposed as Actions, they shall not be browsable. The names of these actions will be communicated to the Client as part of a UID document. These names can be passed to TranslateBrowsePathsToNodeIds to get the matching NodeId.

All Actions are instantiated in both the offline and the online representation of a Device.

The BrowseName Attribute of an Action is the EDD identifier of the METHOD.

The DisplayName Attribute of an Action is the LABEL attribute of the EDDL METHOD.

The Description Attribute of an Action is the HELP attribute of the EDDL METHOD. Bad_AttributeIdInvalid shall be used if the EDD contains no Help.

The execution of an Action causes the execution of the EDD Method implementation. This is specified in detail in IEC 62769-2.

The METHODS specified in an FDI Package may be declared private using the PRIVATE Attribute specified in 61804-3. The FDI Server shall create Action Objects in the Information Model for the private METHODS but they shall not be browsable. The FDI Server shall return the NodeIds of private Actions when the name of such an Action is passed to TranslateBrowsePathsToNodeIds (the startingNode argument shall be the "ActionSet" Object). Once the FDI Client has obtained the NodeId, private Actions can be invoked and processed in the same way as public (browsable) Actions.

15.9 UIPs

UIPs are specified in the FDI Package via the SupportedUIPs element. Each UIP is uniquely identified by a UUID.

UIPs are not exposed in the FDI AddressSpace but they can be referenced and accessed using the UUID. This is specified in IEC 62769-2.

15.10 Protocols, Networks and Connection Points

The FDI Server will create specific Protocol Types (sub-type of the ProtocolType defined in IEC 62541-100) for natively supported Protocols. Additional Protocol Types might be created for Communication Servers based on the Protocols they support.

Specific ConnectionPoint Types (sub-types of the ConnectionPointType defined in IEC 62541-100) will be created for natively supported Protocols. Additional Types might be created for Communication Servers based on the Protocols they support.

The types are created as specified in the protocol-specific series of standards IEC 62769-1xx.

Annex A (normative)

Namespace and Mappings

This appendix defines the numeric identifiers for all of the numeric *NodeIds* defined in this standard. 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.

The *NamespaceUri* <http://fdi-cooperation.com/OpcUa/FDI5/> is applied to *NodeIds* defined here.

The CSV released with this version of the standard can be found here.

http://www.fdi-cooperation.com/tl_files/Specification/1.0/Schemas/Opc.Ua.Fdi5.NodeIds.csv

An electronic version of the complete Information Model defined in this standard is also provided. It follows the XML Information Model Schema syntax defined in IEC 62541-6.

The Information Model Schema released with this version of the standard can be found here:

http://www.fdi-cooperation.com/tl_files/Specification/1.0/Schemas/Opc.Ua.Fdi5.NodeSet2

Bibliography

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

IEC 62541-7, *OPC unified architecture – Part 7: Profiles*

FDI-2021, *FDI Project Technical Specification – Part 1: Overview*
<available at www.fdi-cooperation.com>

FDI-2022, *FDI Project Technical Specification – Part 2: FDI Client*
<available at www.fdi-cooperation.com>

FDI-2023, *FDI Project Technical Specification – Part 3: FDI Server*
<available at www.fdi-cooperation.com>

FDI-2024, *FDI Project Technical Specification – Part 4: FDI Packages*
<available at www.fdi-cooperation.com>

FDI-2025, *FDI Project Technical Specification – Part 5: FDI Information Model*
<available at www.fdi-cooperation.com>

FDI-2026, *FDI Project Technical Specification – Part 6: FDI Technology Mapping*
<available at www.fdi-cooperation.com>

FDI-2027, *FDI Project Technical Specification – Part 7: FDI Communication Devices*
<available at www.fdi-cooperation.com>

NAMUR NE107, *Self-Monitoring and Diagnosis of Field Devices*
<available at www.namur.de>

IECNORM.COM :: Click to view the full PDF of IEC 62769-5:2015

SOMMAIRE

AVANT-PROPOS	62
INTRODUCTION	64
1 Domaine d'application	65
2 Références normatives	66
3 Termes, définitions, abréviations, acronymes et conventions	67
3.1 Termes et définitions	67
3.2 Abréviations et acronymes	67
3.3 Conventions pour la notation graphique	67
4 Vue d'ensemble de l'Architecture Unifiée OPC	69
4.1 Généralités	69
4.2 Vue d'ensemble des Appareils OPC UA	70
5 Concepts	72
5.1 Généralités	72
5.2 Topologie d'appareils	72
5.3 En ligne/Hors ligne	75
5.4 Catalogue (Définitions de Type)	75
5.5 Communication	76
6 Organisation de l'AddressSpace	76
7 Modèle d'Appareil pour FDI	76
7.1 Généralités	76
7.2 En ligne/Hors ligne	77
7.3 Santé de l'Appareil	77
7.4 Eléments de l'interface utilisateur	78
7.4.1 Généralités	78
7.4.2 Type UI Description	78
7.4.3 Type UI Plug-in	79
7.5 Informations de support spécifique au type	81
7.6 Actions	81
7.6.1 Vue d'ensemble	81
7.6.2 Type Action	83
7.6.3 Type ActionService	83
7.6.4 Objet ActionService	84
7.6.5 Méthode InvokeAction	84
7.6.6 Méthode RespondAction	85
7.6.7 Méthode AbortAction	86
8 Réseau et connectivité	86
9 Fonctions utilitaires	87
9.1 Vue d'ensemble	87
9.2 Locking	87
9.3 EditContext	87
9.3.1 Vue d'ensemble	87
9.3.2 EditContext Type	88
9.3.3 Objet EditContext	88
9.3.4 Méthode GetEditContext	89
9.3.5 Méthode RegisterNodes	89

9.3.6	Méthode Apply.....	91
9.3.7	Méthode Reset	92
9.3.8	Méthode Discard	92
9.4	Direct Device Access	93
9.4.1	Généralités	93
9.4.2	Type DirectDeviceAccess	93
9.4.3	Objet DirectDeviceAccess.....	94
9.4.4	Méthode InitDirectAccess	95
9.4.5	Méthode EndDirectAccess	95
9.4.6	Méthode Transfer	96
10	Types Parameter	97
10.1	Généralités	97
10.2	Propriété ScalingFactor	98
10.3	Propriété Min_Max_Values	98
11	StatusCodes FDI	99
12	Éléments de topologie spécialisés	99
13	Vérification	99
13.1	Généralités	99
13.2	Informations de contexte fournies par le Client FDI	100
13.3	Méthode LogAuditTrailMessage	100
14	Version de Serveur FDI	101
15	Mapping des informations de Paquetage FDI au Modèle d'Information FDI	101
15.1	Généralités	101
15.2	Localisation	102
15.2.1	Texte localisé	102
15.2.2	Unités techniques	102
15.3	Appareil	102
15.3.1	Généralités	102
15.3.2	Mapping des Attributs à un Nœud DeviceType spécifique	102
15.3.3	Mapping aux propriétés	102
15.3.4	Mapping à ParameterSet	103
15.3.5	Mapping aux Groupes Fonctionnels	103
15.3.6	Mapping à DeviceTypeImage	103
15.3.7	Mapping à Documentation	103
15.3.8	Mapping avec ProtocolSupport	103
15.3.9	Mapping à ImageSet	103
15.3.10	Mapping à ActionSet	104
15.3.11	Mapping à MethodSet	104
15.4	Blocs	104
15.4.1	Généralités	104
15.4.2	Mapping aux Attributs	104
15.4.3	Mapping à ParameterSet	104
15.4.4	Mapping aux Groupes Fonctionnels	104
15.4.5	Mapping à ActionSet	105
15.4.6	Mapping à MethodSet	105
15.4.7	Règles d'instanciation	105
15.5	Paramètre	105
15.5.1	Généralités	105

15.5.2	Paramètres privés	109
15.5.3	MIN_Value et MAX_Value.....	109
15.5.4	Unités techniques	109
15.5.5	Paramètres énumérés	109
15.5.6	Paramètres Bit-enumerated	110
15.5.7	Représentation des enregistrements.....	110
15.5.8	Représentation des matrices et listes des Paramètres avec types de données simples.....	111
15.5.9	Représentation des matrices de valeurs et des listes de Paramètres RECORD	111
15.5.10	Représentation de COLLECTION et REFERENCE ARRAY	112
15.5.11	SCALING_FACTOR.....	112
15.6	Groupes Fonctionnels	113
15.7	Éléments AXIS dans les UID.....	113
15.8	Actions	114
15.9	UIP	114
15.10	Protocoles, Réseaux et Points de Connexion.....	114
Annexe A (informative) Espace de noms et Mappings.....		115
Bibliographie.....		116
Figure 1	– Diagramme de l'architecture FDI.....	66
Figure 2	– Notation graphique de l'OPC UA pour les NodeClasses	68
Figure 3	– Notation graphique de l'OPC UA pour les références	68
Figure 4	– Exemple de notation graphique de l'OPC UA.....	69
Figure 5	– Référence de Type optimisée.....	69
Figure 6	– Exemple d'Appareils OPC UA: Groupes Fonctionnels	71
Figure 7	– Exemple d'Appareils OPC UA: Composants configurables	72
Figure 8	– Exemple de système d'automatisation.....	73
Figure 9	– Exemple de Topologie d'un appareil.....	74
Figure 10	– Exemple de Types d'Appareils représentant un catalogue	76
Figure 11	– Composant en ligne pour l'accès aux données d'appareil.....	77
Figure 12	– Hiérarchie des Types UserInterface	78
Figure 13	– Intégration des Actions au sein de TopologyElement.....	82
Figure 14	– Service Action	84
Figure 15	– Type et instance EditContext.....	89
Figure 16	– DirectDeviceAccessType.....	94
Figure 17	– Instance DirectDeviceAccess	94
Figure 18	– VariablesTypes de l'OPC UA y compris DataAccess OPC UA	97
Figure 19	– Exemple: Variable complexe représentant un RECORD	110
Figure 20	– Variable complexe représentant un VALUE_ARRAY de plusieurs RECORD	112
Tableau 1	– Définition du UIDescriptionType	79
Tableau 2	– Définition du UIPlugInType	79
Tableau 3	– TopologyElementType avec suppléments pour les Actions	82
Tableau 4	– FunctionalGroupType avec suppléments pour les Actions	83
Tableau 5	– Définition du ActionType.....	83

Tableau 6 – Définition du ActionServiceType	83
Tableau 7 – Arguments de la Méthode InvokeAction	85
Tableau 8 – Définition de l'AddressSpace de la Méthode InvokeAction	85
Tableau 9 – Arguments de la Méthode RespondAction	85
Tableau 10 – Définition de l'AddressSpace de la Méthode RespondAction	86
Tableau 11 – Arguments de la Méthode AbortAction	86
Tableau 12 – Définition de l'AddressSpace de la Méthode AbortAction	86
Tableau 13 – Définition du EditContextType	88
Tableau 14 – Arguments de la Méthode GetEditContext	89
Tableau 15 – Définition de l'AddressSpace de la Méthode GetEditContext	89
Tableau 16 – Arguments de la Méthode RegisterNodes	90
Tableau 17 – Définition de l'AddressSpace de la Méthode RegisterNodes	90
Tableau 18 – Structure du type de données RegistrationParameters	90
Tableau 19 – Structure du type de données RegisterNodesResult	91
Tableau 20 – Arguments de la Méthode Apply	91
Tableau 21 – Définition de l'AddressSpace de la Méthode Apply	91
Tableau 22 – Structure du DataType ApplyResult	92
Tableau 23 – Arguments de la Méthode Reset	92
Tableau 24 – Définition de l'AddressSpace de la Méthode Reset	92
Tableau 25 – Arguments de la Méthode Discard	93
Tableau 26 – Définition de l'AddressSpace de la Méthode Discard	93
Tableau 27 – Définition du DirectDeviceAccessType	94
Tableau 28 – Définition de l'instance DirectDeviceAccess	95
Tableau 29 – Arguments de la Méthode InitDirectAccess	95
Tableau 30 – Définition de l'AddressSpace de la Méthode InitDirectAccess	95
Tableau 31 – Arguments de la Méthode EndDirectAccess	96
Tableau 32 – Définition de l'AddressSpace de la Méthode EndDirectAccess	96
Tableau 33 – Arguments de la Méthode Transfert	96
Tableau 34 – Définition de l'AddressSpace de la Méthode Transfert	96
Tableau 35 – Définition de la Propriété ScalingFactor	98
Tableau 36 – Définition de la Propriété Min_Max_Values	98
Tableau 37 – Structure du type de données Variant_Range	98
Tableau 38 – Définition de Variant_Range	98
Tableau 39 – Codes de résultat du niveau opérationnel Good	99
Tableau 40 – Arguments de la Méthode LogAuditTrailMessage	100
Tableau 41 – Définition de l'AddressSpace de la Méthode LogAuditTrailMessage	101
Tableau 42 – Définition de la Propriété FDIServerVersion	101
Tableau 43 – Mapping des propriétés DeviceType	102
Tableau 44 – Mise en place des Attributs Variable de l'OPC UA à partir des attributs de variable de l'EDDL	106
Tableau 45 – Correspondance entre les types de données normalisés de l'EDDL et de l'OPC UA	106

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

INTÉGRATION DES APPAREILS DE TERRAIN (FDI) –

Partie 5: Modèle d'Information FDI

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. A 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.

La Norme internationale IEC 62769-5 a été établie par le sous-comité 65E: Les appareils 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.

Le texte de cette norme est issu des documents suivants:

CDV	Rapport de vote
65E/348/CDV	65E/425/RVD

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

Cette publication a été rédigée selon les Directives ISO/IEC, Partie 2.

Une liste de toutes les parties de la série IEC 62769, publiées sous le titre général *Intégration des appareils de terrain (FDI)*, peut être consultée sur le site web de l'IEC.

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de stabilité indiquée sur le site web de l'IEC sous "<http://webstore.iec.ch>" dans les données relatives à la publication recherchée. A cette date, la publication sera

- reconduite,
- supprimée,
- remplacée par une édition révisée, ou
- amendée.

IMPORTANT – Le logo "colour inside" qui se trouve sur la page de couverture de cette publication 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

La Commission Electrotechnique Internationale (IEC) attire l'attention sur le fait qu'il est déclaré que la conformité avec les dispositions du présent document peut impliquer l'utilisation de brevets intéressants:

- a) la méthode de fourniture et d'installation des fonctionnalités spécifiques aux appareils (cf. famille de brevets DE10357276);
- b) la méthode et l'appareil utilisés pour l'accès à un module fonctionnel du système d'automation (cf. famille de brevets EP2182418);
- c) les méthodes et les appareils utilisés pour diminuer les exigences mémoire relatives aux applications logicielles du système de commande de processus (cf. famille de brevets US2013232186);
- d) modèle d'objet d'appareil extensible (cf. famille de brevets US12/893,680).

L'IEC ne prend pas position quant à la preuve, à la validité et à la portée de ces droits de propriété.

Le détenteur de ces droits de propriété a donné l'assurance à l'IEC qu'il consent à négocier des licences avec des demandeurs du monde entier, soit sans frais soit à des termes et conditions raisonnables et non discriminatoires. A ce propos, la déclaration du détenteur des droits de propriété est enregistrée à l'IEC. Des informations peuvent être demandées à:

- a) ABB Research Ltd
Claes Ryttoft
Affolterstrasse 4
Zurich, 8050
Suisse
- b) Phoenix Contact GmbH & Co KG
Intellectual Property, Licenses & Standards
Flachsmarktstrasse 8, 32825 Blomberg
Allemagne
- c) Fisher Controls International LLC
John Dilger, Emerson Process Management LLLP
301 S. 1st Avenue, Marshalltown, Iowa 50158
Etats-Unis
- d) Rockwell Automation Technologies, Inc.
1 Allen-Bradley Drive
Mayfield Heights, Ohio 44124
Etats-Unis

L'attention est d'autre part attirée sur le fait que certains des éléments du présent document peuvent faire l'objet de droits de propriété autres que ceux qui ont été mentionnés ci-dessus. L'IEC ne saurait être tenue pour responsable de l'identification de ces droits de propriété en tout ou partie.

L'ISO (www.iso.org/patents) et l'IEC (<http://patents.iec.ch>) tiennent à jour des bases de données en ligne sur les brevets relatifs à leurs normes. Les utilisateurs sont encouragés à consulter ces bases de données pour obtenir l'information la plus récente concernant les brevets.

INTÉGRATION DES APPAREILS DE TERRAIN (FDI) –

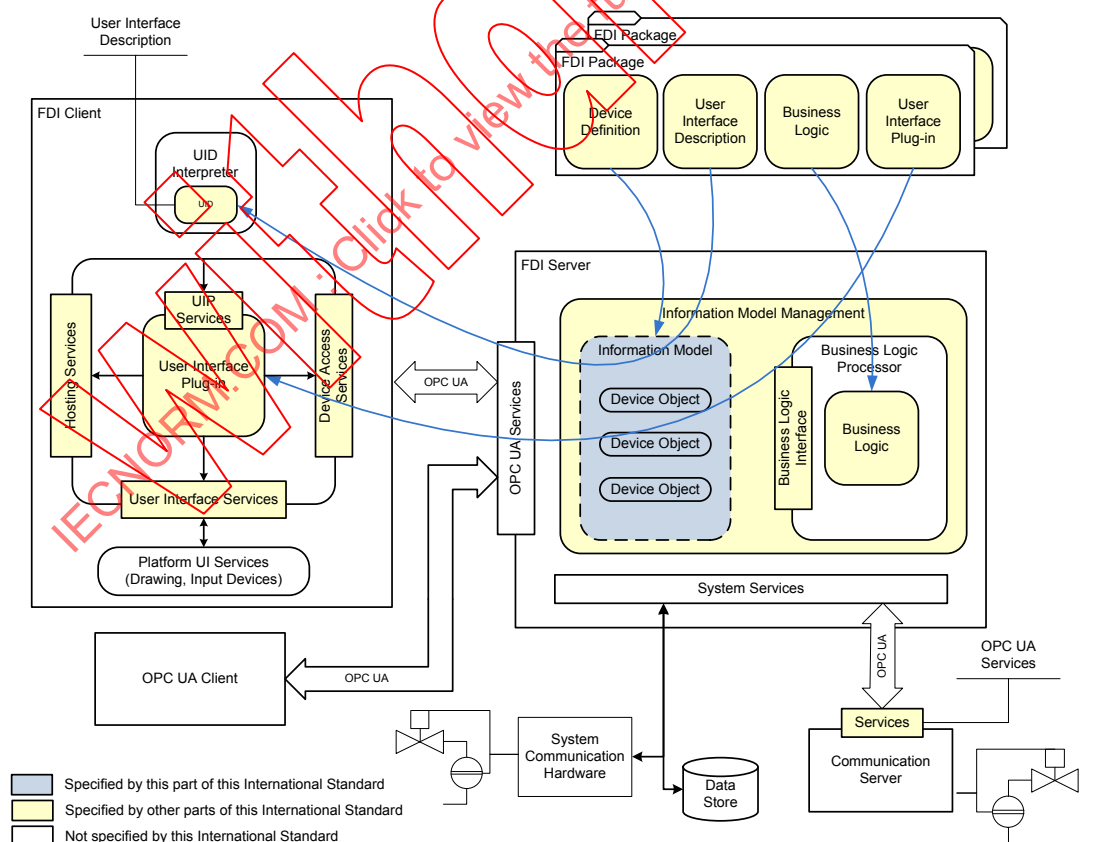
Partie 5: Modèle d'Information FDI

1 Domaine d'application

La présente partie de l'IEC 62769 définit le Modèle d'Information FDI. L'un des principaux objectifs du Modèle d'Information est de refléter la topologie du système d'automatisation. Il représente les appareils du système d'automatisation ainsi que les réseaux de communication connectés, y compris leurs propriétés, leurs relations et les opérations dont ils peuvent faire l'objet. Les types présents dans l'AddressSpace du Serveur FDI constituent un catalogue, construit à partir des FDI Packages (Paquetages FDI).

Les types fondamentaux pour le Modèle d'Information FDI sont définis dans l'OPC UA pour les Appareils (IEC 62541-100). Le Modèle d'Information FDI spécifie des extensions pour quelques cas spéciaux et explique la façon dont ces types sont utilisés et dont les contenus sont construits à partir des éléments de DevicePackages.

L'architecture FDI complète est présentée à la Figure 1. Les composants architecturaux relevant du domaine d'application du présent document ont été mis en évidence dans cette illustration.



IEC

Anglais	Français
User Interface Description	Description d'Interface Utilisateur
FDI Client	Client FDI

Anglais	Français
UID Interpreter	Interprète UID
UID	UID
UIP Services	Services UIP
Hosting Services	Services d'Hébergement
User Interface Plug-in	Plugiciel d'Interface Utilisateur
Device Access Services	Services d'Accès à l'Appareil
User Interface Services	Services d'Interface Utilisateur
Platform UI Services (Drawing, Input Devices)	Services d'UI de Plate-Forme (Dessin, Appareils d'Entrée)
OPC UA Client	Client OPC UA
FDI Package	Paquetage FDI
Device Definition	Définition d'Appareil
Business Logic	Logique Appllicative
FDI Server	Serveur FDI
OPC UA Services	Services OPC UA
Information Model Management	Gestion du Modèle d'Information
Information Model	Modèle d'Information
Device Object	Objet Device (Appareil)
Business Logic Processor	Processeur de Logique Appllicative
Business Logic Interface	Interface de Logique Appllicative
System Services	Services Système
System Communication Hardware	Matériel de Communication du Système
Data Store	Magasin de Données
Communication Server	Serveur de Communication
Specified by this part of this International Standard	Spécifié par la présente partie de la présente Norme internationale
Specified by other parts of this International Standard	Spécifié par d'autres parties de la présente Norme internationale
Not specified by this International Standard	Non spécifié par la présente Norme internationale

Figure 1 – Diagramme de l'architecture FDI

2 Références normatives

Les documents suivants sont cités en référence de manière normative, en intégralité ou en partie, dans le présent document et sont indispensables pour son application. 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 61784-1, *Réseaux de communication industriels – Profils – Partie 1: Profils de bus de terrain*

IEC 61804-3¹, *Blocs fonctionnels (FB) pour les procédés industriels et le Langage de description électronique de produit (EDDL) – Partie 3: Sémantique et syntaxe EDDL*

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

IEC 62541-4, *Architecture unifiée OPC – Partie 4: Services*

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

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

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

IEC 62541-100², *Architecture unifiée OPC – Partie 100: OPC UA pour Appareils*

IEC 62769-1, *Intégration des appareils de terrain (FDI) – Partie 1: Vue d'ensemble*

NOTE L'IEC 62769-1 est techniquement identique à la FDI-2021.

IEC 62769-2, *Intégration des appareils de terrain (FDI) – Partie 2: Client FDI*

NOTE L'IEC 62769-2 est techniquement identique à la FDI-2022.

IEC 62769-4, *Intégration des appareils de terrain (FDI) – Partie 4: Paquetages FDI*

NOTE L'IEC 62769-4 est techniquement identique à la FDI-2024.

IEC 62769-7, *Intégration des appareils de terrain (FDI) – Partie 7: Appareils de Communication FDI*

NOTE L'IEC 62769-7 est techniquement identique à la FDI-2027.

3 Termes, définitions, abréviations, acronymes et conventions

3.1 Termes et définitions

Pour les besoins du présent document, les termes et définitions donnés dans l'IEC 62769-1 s'appliquent.

3.2 Abréviations et acronymes

Pour les besoins du présent document, les abréviations et acronymes de l'IEC 62769-1, ainsi que les suivants s'appliquent.

IHM	Interface Homme Machine (HMI – Human Machine Interface)
SCADA	Supervisory Control and Data Acquisition (Système de supervision, contrôle et acquisition de données)
TCP	Transmission Control Protocol (Protocole de contrôle de transmission)

3.3 Conventions pour la notation graphique

L'OPC UA définit une notation graphique pour un AddressSpace de l'OPC UA. Elle définit des symboles graphiques pour tous les NodeClasses et pour la façon les différents types de

¹ A paraître.

² A l'étude.

références entre les Nodes peuvent être visualisés La Figure 2 présente les symboles pour les NodeClasses utilisés dans la présente norme. Les types représentant des NodeClasses comportent toujours une ombre.

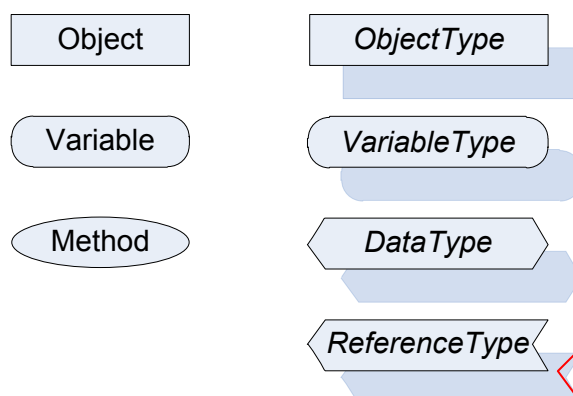
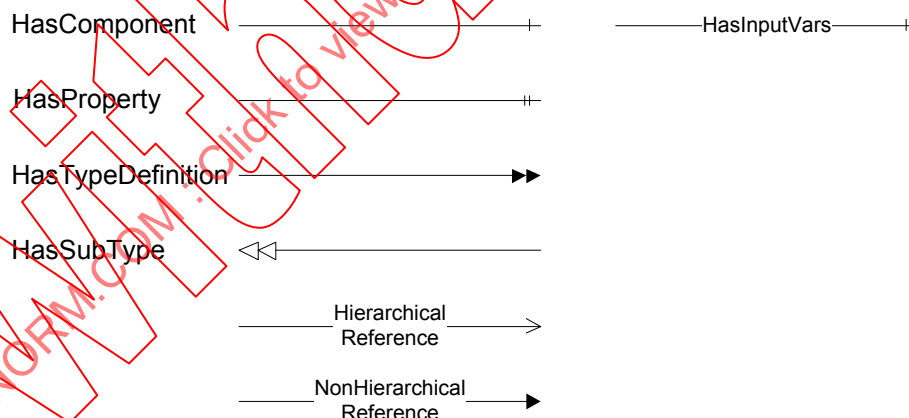


Figure 2 – Notation graphique de l'OPC UA pour les NodeClasses

La Figure 3 présente les symboles pour les ReferenceTypes utilisés dans la présente norme. Le symbole de Référence pointe généralement du Nœud d'origine vers le Nœud cible. La seule exception est la Référence HasSubType. Les Références les plus importantes, telles que HasComponent, HasProperty, HasTypeDefinition et HasSubType, comportent des symboles spécifiques permettant de ne pas indiquer le nom de la Référence. Pour les autres ReferenceTypes ou ReferenceTypes dérivés, le nom de la ReferenceType est utilisé avec le symbole.



IEC

Anglais	Français
Hierarchical Reference	Référence hiérarchique
NonHierarchical Reference	Référence non hiérarchique

Figure 3 – Notation graphique de l'OPC UA pour les références

La Figure 4 présente un exemple typique de l'utilisation de la notation graphique. Object_A et Object_B sont des instances d'ObjectType_Y indiquées par les Références HasTypeDefinition. L'ObjectType_Y est dérivé de l'ObjectType_X par la Référence HasSubType. L'Object_A comporte les composants Variable_1, Variable_2 et Method_1.

Afin de décrire les composants d'un Object sur l'ObjectType, les mêmes NodeClasses et Références sont utilisées sur l'Object et sur l'ObjectType, comme indiqué dans l'exemple pour

l'ObjectType_Y. Les Nodes utilisés pour décrire un ObjectType sont des Nodes de déclaration d'instance.

Afin de fournir des informations plus détaillées pour un Node, un sous-ensemble ou l'ensemble des Attributs et leurs valeurs peuvent être ajoutés à un symbole graphique (voir, par exemple, Variable_1, le composant de l'Object_A à la Figure 4).

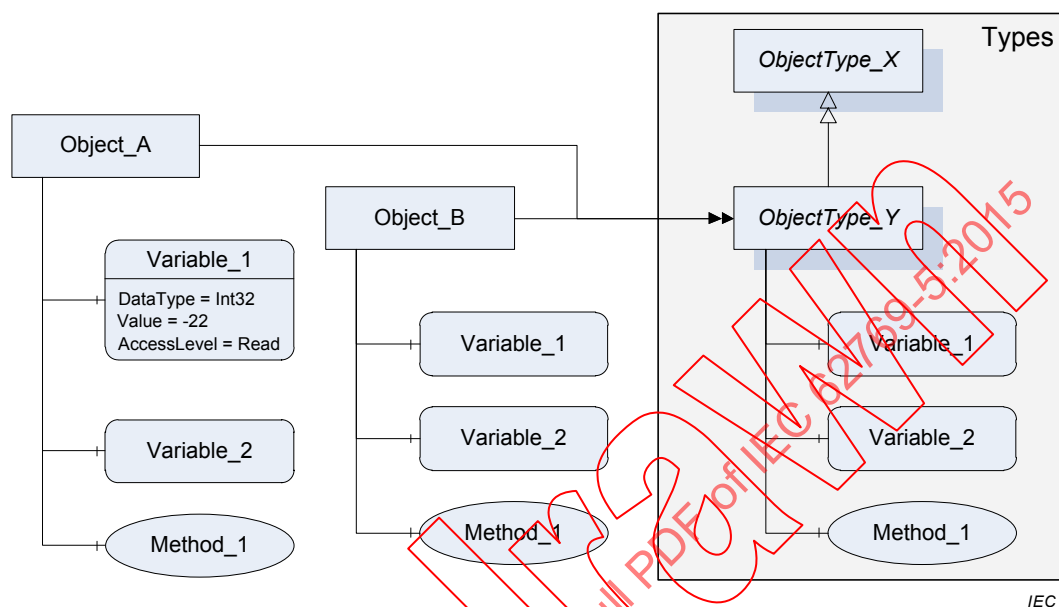


Figure 4 – Exemple de notation graphique de l'OPC UA

Afin d'améliorer la lisibilité, le présent document inclut fréquemment le nom de type à l'intérieur de la boîte d'instance, plutôt que d'afficher les deux boîtes et la référence entre celles-ci. Cette optimisation est représentée à la Figure 5.

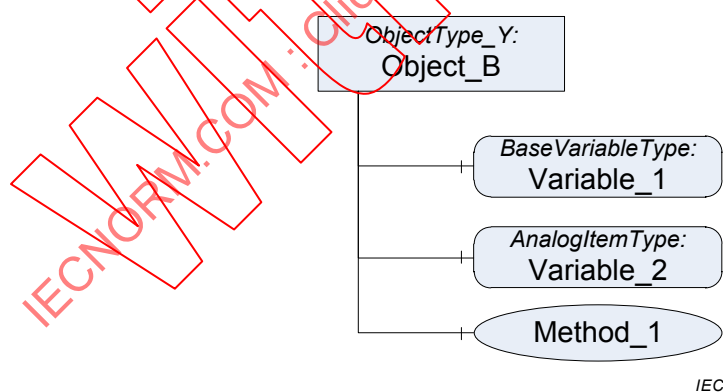


Figure 5 – Référence de Type optimisée

4 Vue d'ensemble de l'Architecture Unifiée OPC

4.1 Généralités

Le principal cas d'utilisation pour les normes OPC est l'échange de données en ligne entre des appareils et des systèmes IHM ou SCADA. Dans ce cas d'utilisation, les données de l'appareil sont fournies par un serveur OPC et sont consommées par un client OPC intégré dans le système IHM ou SCADA. L'OPC fournit une fonctionnalité permettant de naviguer

dans des espaces de nom hiérarchiques contenant des éléments de données afin de lire, d'écrire et d'examiner ces éléments pour les modifications de données.

L'OPC UA comprend des fonctionnalités telles que l'accès aux données, les alarmes et les données historiques, à travers des mécanismes de communication indépendants de la plateforme et des capacités de modélisation génériques, extensibles et orientées objet, pour les informations qu'un système veut exposer.

La version actuelle de l'OPC UA définit un protocole TCP binaire optimisé pour la communication intranet haute performance ainsi qu'un mapping aux Services Web. Le modèle de service abstrait ne dépend pas d'un mapping à un protocole spécifique et permet d'ajouter de nouveaux protocoles ultérieurement. Des fonctionnalités telles que la sécurité, le contrôle de l'accès et la fiabilité sont directement incluses dans les mécanismes de transport. Sur la base de l'indépendance des protocoles vis-à-vis de la plateforme, les serveurs et les clients OPC UA peuvent être directement intégrés dans les appareils et les contrôleurs.

Le modèle d'information OPC UA apporte aux serveurs une méthode normalisée pour exposer des objets aux clients. Dans les termes de l'OPC UA, les objets sont composés d'autres objets, de variables et de méthodes. L'OPC UA permet également d'exprimer des relations avec d'autres objets.

L'ensemble des objets et des informations connexes qu'un serveur OPC UA met à la disposition des clients est appelé son AddressSpace. Les éléments d'un modèle d'objet OPC UA sont représentés dans l'AddressSpace par un ensemble de Nœuds décrits par des attributs et interconnectés par des références. L'OPC UA définit diverses classes de Nœuds pour représenter les composants de l'AddressSpace, surtout les Objects, Variables, Methods, ObjectTypes, DataTypes et ReferenceTypes. Chaque NodeClass possède un ensemble défini d'attributs.

Les Objets (Objects) sont utilisés pour représenter des composants tels que des dossiers, des appareils ou des réseaux. Un Objet est associé à un ObjectType correspondant qui fournit les définitions de cet Objet.

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

Les Propriétés (Properties) sont des caractéristiques définies par un serveur pour les Objets, les DataVariables et d'autres Nœuds. Il n'est pas permis de définir des Propriétés pour d'autres Propriétés. La Propriété Manufacturer d'un appareil est un exemple de Propriété d'un Objet.

Les DataVariables représentent le contenu d'un Objet. Les DataVariables peuvent avoir des DataVariables comme composants. Elles sont utilisées généralement par les Serveurs pour exposer des éléments individuels de matrices et de structures. La présente norme utilise les DataVariables principalement pour représenter les Paramètres des Appareils.

4.2 Vue d'ensemble des Appareils OPC UA

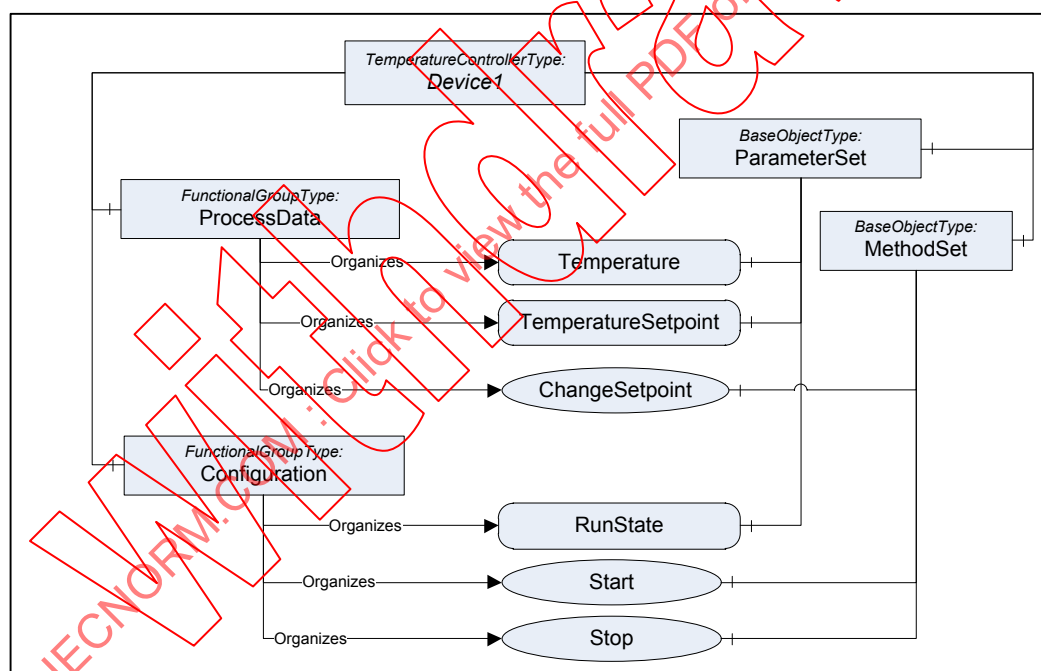
La norme Architecture unifiée OPC pour Appareils (IEC 62541-100) est une extension de la série globale de normes pour l'Architecture Unifiée OPC et définit les modèles d'information associés aux appareils. L'IEC 62541-100 décrit trois modèles qui s'appuient les uns sur les autres, comme suit:

- Le Modèle d'appareil (de base) a pour but de fournir une vue unifiée des appareils, quels que soient les protocoles d'appareil sous-jacents.
- Le Modèle de Communication d'Appareil ajoute des éléments d'information de Réseau et de Connexion afin d'être en mesure de créer les topologies de communication.

- Enfin, le Modèle Device Integration Host ajoute des éléments et des règles supplémentaires nécessaires pour que les systèmes hôtes gèrent l'intégration pour un système complet. Il permet également de refléter la topologie du système d'automatisation avec les appareils ainsi qu'avec les réseaux de communication connectés.

Le modèle d'information des Appareils spécifie les différents ObjectTypes et d'autres éléments de l'AddressSpace utilisés pour représenter les Appareils et les composants liés, comme l'infrastructure de communication dans un AddressSpace (Espace d'adressage) de l'OPC UA. Les principaux cas d'utilisation sont la configuration et le diagnostic d'Appareil, mais il permet d'apporter une méthode générale et normalisée pour permettre aux applications d'accéder aux informations liées à un Appareil.

La Figure 6 présente un exemple de contrôleur de température représenté sous forme d'Objet Appareil. Le composant ParameterSet contient toutes les Variables décrivant l'Appareil. Le composant MethodSet contient toutes les Méthodes fournies par l'Appareil. Les deux composants sont hérités du TopologyElementType, qui est le type Objet racine dans la hiérarchie de type d'Objet d'Appareil. Les Objets du type FunctionalGroupType sont utilisés pour regrouper les Paramètres et les Méthodes de l'Appareil dans des groupes logiques. Le FunctionalGroupType et le concept de regroupement sont définis dans l'IEC 62541-100, mais les groupes sont spécifiques au DeviceType, c'est-à-dire que les groupes ProcessData et Configuration sont définis par le TemperatureControllerType dans cet exemple.

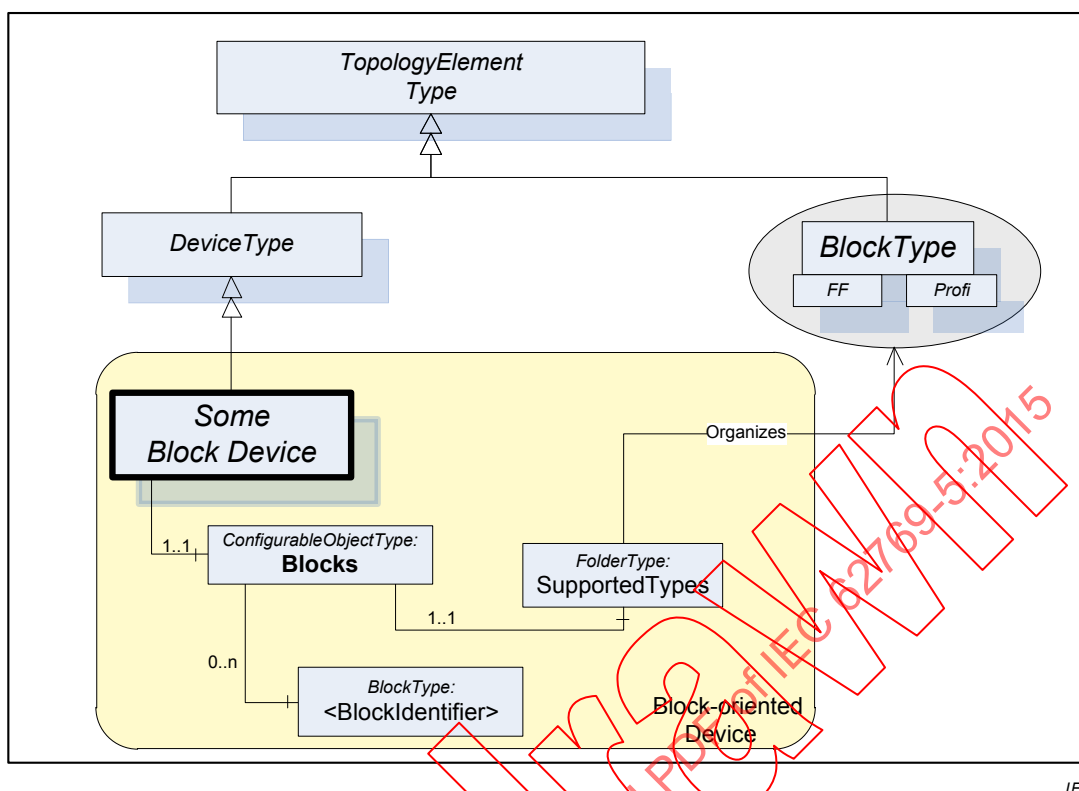


IEC

Anglais	Français
Organizes	Organise

Figure 6 – Exemple d'Appareils OPC UA: Groupes Fonctionnels

Un autre concept de l'IEC 62541-100 est représenté à la Figure 7. Le ConfigurableObjectType est utilisé pour fournir une méthode pour grouper des sous-composants d'un Appareil et pour indiquer quels types de sous-composants peuvent être instanciés. Les types autorisés sont référencés depuis le dossier SupportedTypes. Ces informations peuvent être utilisées par des clients de configuration pour permettre à un utilisateur de sélectionner le type à instancier en tant que sous-composant d'un Appareil.



IEC

Anglais	Français
Block-oriented Device	Appareil orienté bloc
Some Block Device	Certains appareils bloc

Figure 7 – Exemple d'Appareils OPC UA: Composants configurables

Le dossier SupportedTypes peut contenir différents sous-ensembles d'ObjectTypes pour différentes instances de l'Appareil orienté Bloc, en fonction de leur configuration actuelle, puisque la liste contient uniquement des types pouvant être instanciés pour la configuration actuelle.

5 Concepts

5.1 Généralités

Le Serveur FDI fournit aux Clients FDI l'accès aux informations sur les instances d'Appareil et les types d'Appareils, peu importe où sont stockées les informations, par exemple, dans l'Appareil lui-même ou dans un magasin de données. Ces informations sont fournies via les Services OPC UA et sont appelées Modèle d'Information FDI.

Le Modèle d'Information FDI spécifie les entités auxquelles il est permis d'accéder dans le Serveur FDI, y compris leurs propriétés, leurs relations et les différentes opérations dont elles peuvent faire l'objet. Le fait de savoir quels types d'Appareil ou autres éléments topologiques sont disponibles dans un Serveur FDI donné dépend en grande partie des informations contenues dans les Paquetages FDI.

5.2 Topologie d'appareils

L'un des principaux objectifs du Modèle d'Information est de refléter la topologie du système d'automatisation. Le Modèle d'Information représente donc les Appareils du système d'automatisation ainsi que les réseaux de communication de connexion. Le point d'entrée Topologie d'Appareil est le point de départ au sein du Modèle d'Information pour la topologie

du système d'automatisation. Le point d'entrée Appareils de Communication contient les appareils de communication utilisés par le Serveur FDI pour accéder aux éléments de la topologie. La Figure 8 et la Figure 9 illustrent un exemple de configuration et de la topologie configurée telle qu'elle apparaîtra dans l'AddressSpace du Serveur FDI (les détails sont omis).

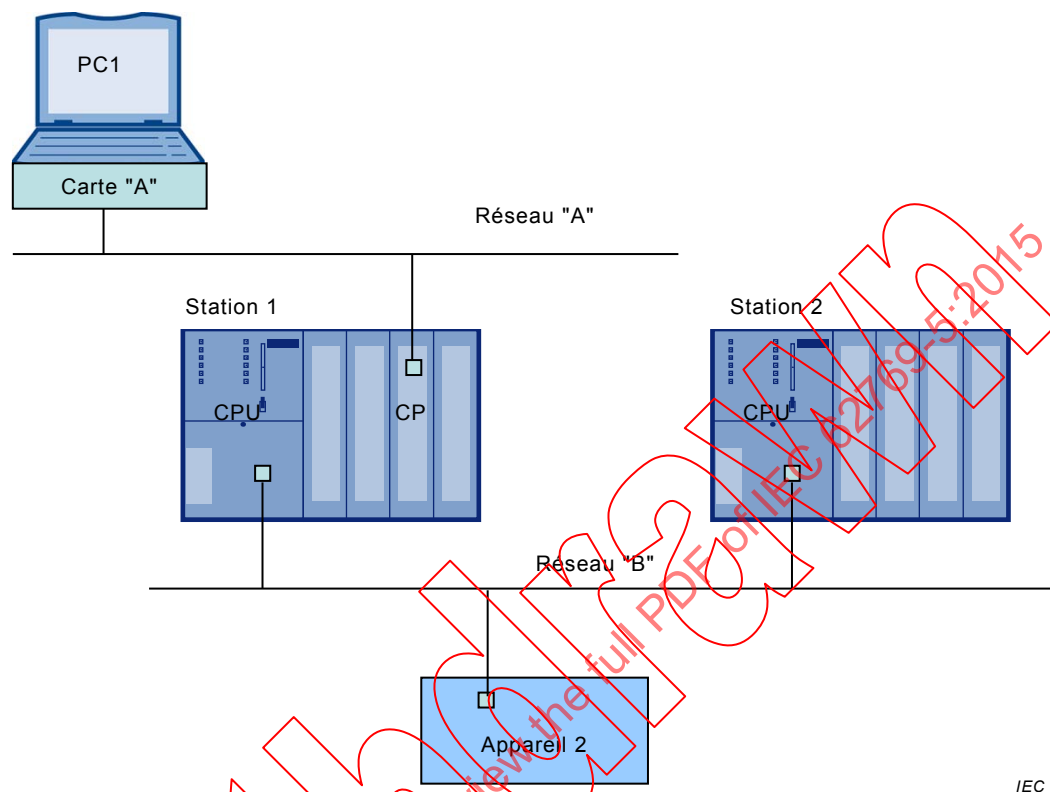
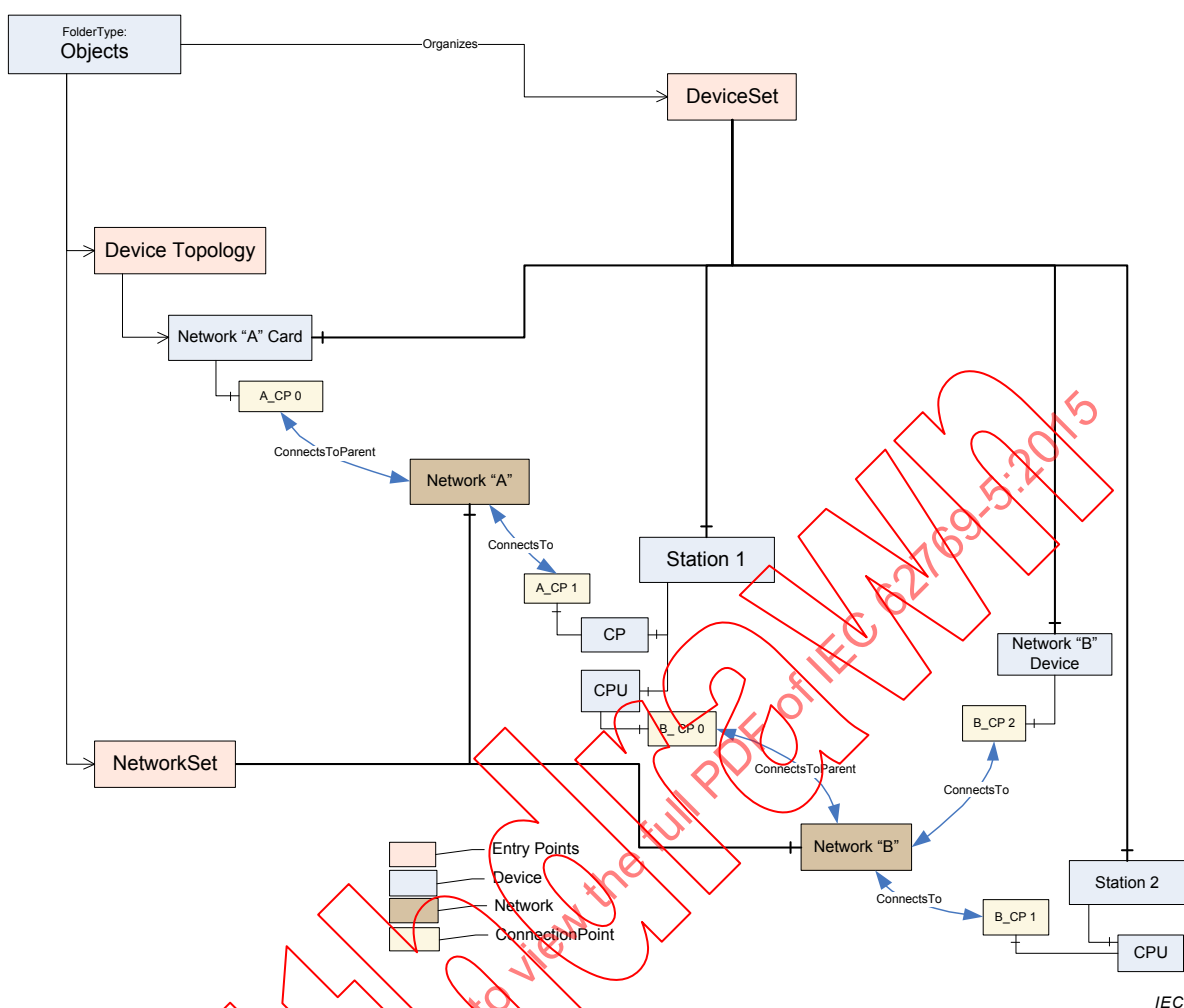


Figure 8 – Exemple de système d'automatisation

Le PC dans la Figure 8 représente le boîtier du Serveur FDI. Le Serveur FDI communique avec les appareils connectés au Réseau "A" par une Communication Native, et il communique avec les appareils connectés au Réseau "B" par une Communication Imbriquée.



Anglais	Français
Organizes	Organise
Device Topology	Topologie d'Appareil
Network "A" Card	Carte Réseau "A"
Network "A"	Réseau "A"
Network "B"	Réseau "B"
Station 1	Station 1
Station 2	Station 2
Entry Points	Points d'entrée
Device	Appareil
Network	Réseau
ConnectionPoint	Point de Connexion

Figure 9 – Exemple de Topologie d'un appareil

Des rectangles de couleur sont utilisés afin de reconnaître facilement les différents types d'information.

Les rectangles marron représentent les réseaux. Les rectangles bleu clair représentent les Appareils et le jaune clair est utilisé pour les Points de Connexion.

Les rectangles rose clair représentent les points d'entrée qui assurent un comportement commun à travers différentes mises en œuvre:

- DeviceTopology: Nœud de départ pour la configuration de la topologie.
- DeviceSet: Tous les Appareils instanciés sont des composants de cet Objet, c'est-à-dire qu'ils existent dans l'AddressSpace indépendamment de la Topologie d'Appareil.
- NetworkSet: Tous les réseaux sont des composants de cet Objet.

5.3 En ligne/Hors ligne

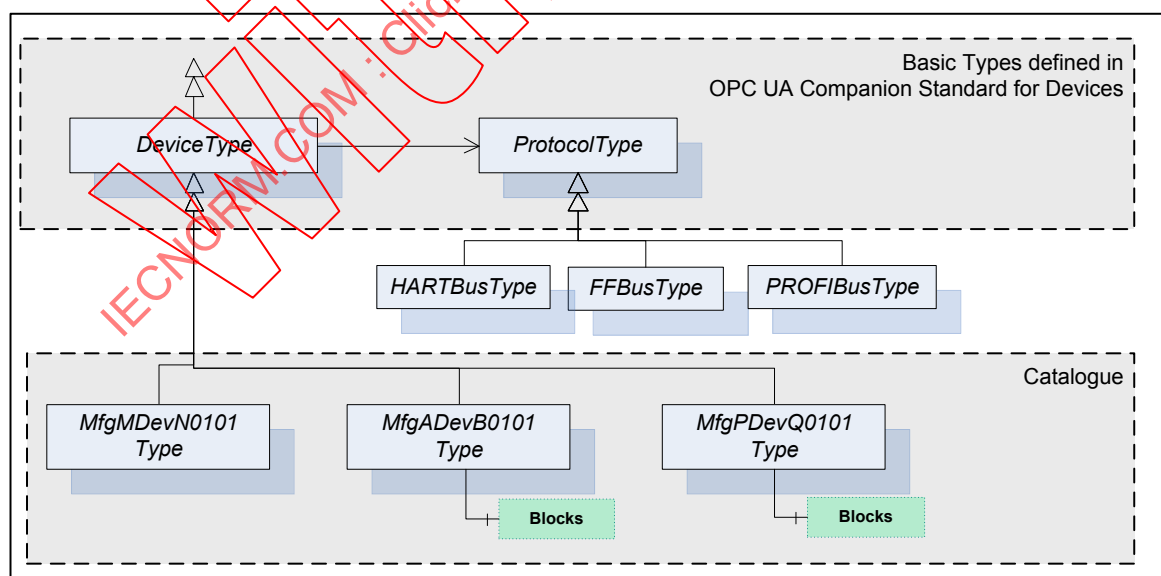
La gestion de la Topologie d'Appareil est une tâche de configuration, c'est-à-dire que les éléments dans la topologie (Appareils, Réseaux et Points de connexion) sont généralement configurés "hors ligne" et seront validés, ultérieurement, sur la base de leur représentation physique dans un réseau réel.

Afin de favoriser l'accès explicite aux informations en ligne ou hors ligne, chaque élément est représenté par deux instances schématiquement identiques, c'est-à-dire qu'il existe un ParameterSet, des FunctionalGroups, et ainsi de suite. Une Référence relie les représentations en ligne et hors ligne et permet de naviguer entre elles.

5.4 Catalogue (Définitions de Type)

Les (sous-ensembles de) TopologyElements pris en charge sont organisés en tant que définitions de Type dans l'AddressSpace OPC UA, formant ainsi une sorte de catalogue. Ces définitions sont généralement générées à partir d'informations descriptives provenant des Paquetages FDI. Les définitions de Type contiennent les Paramètres et les valeurs par défaut des Paramètres, Méthodes, Actions et Groupes Fonctionnels, y compris les éléments de l'interface utilisateur. Le Serveur FDI peut inclure des dossiers dans le modèle de type, afin d'organiser les types selon le fabricant ou d'autres critères.

Les définitions de type peuvent alors être utilisées pour créer des instances d'Appareil dans l'AddressSpace OPC UA. Les instances peuvent être créées soit hors ligne, soit selon des données déterminées par un Balayage. La Figure 10 présente un exemple de certaines définitions de Type (les détails sont omis) telles qu'elles peuvent exister dans l'AddressSpace.



IEC

Anglais	Français
Basic Types defined in OPC UA Companion Standard for Devices	Types de base définis dans la Norme compagne OPC UA pour les Appareils
Catalogue	Catalogue
Blocks	Blocs

Figure 10 – Exemple de Types d'Appareils représentant un catalogue

5.5 Communication

Afin d'intégrer des Appareils, le Serveur FDI a besoin d'être capable de communiquer avec eux. Cela peut être effectué en utilisant une Communication Native ou une Communication Imbriquée.

L'exemple donné dans la Figure 9 spécifie que le Serveur FDI a un accès direct au Réseau PROFINET en utilisant sa carte réseau PROFINET. Afin d'accéder à la "Station 2", le Serveur FDI doit passer par la Station 1 qui fournit les services de communication pour le réseau PROFIBUS DP (voir l'IEC 61784-1, CPF 3). Cela peut être effectué avec une Communication Native, qui est spécifiée dans l'IEC 62769-4. Les Appareils de Communication et les Serveurs de Communication sont spécifiés dans l'IEC 62769-7.

6 Organisation de l'AddressSpace

Afin de favoriser l'interopérabilité des Clients FDI et des Serveurs FDI, un ensemble d'Objets et des relations sont définis dans les articles suivants. Les Serveurs FDI peuvent mettre en œuvre un sous-ensemble de ces Nœuds normalisés, en fonction de leurs capacités.

Selon les règles de l'OPC UA, un Serveur OPC UA sépare l'AddressSpace en deux parties:

- a) La partie "Types" contient les informations relatives à tous les composants qui ont été générés en fonction des informations descriptives provenant des Paquetages FDI (voir 5.4).
- b) La partie "Objets" contient la Topologie d'Appareil avec tous les composants instanciés. Toutes les instances de l'AddressSpace sont reliées à un type du dossier Types.
 - Les points d'entrée DeviceSet, NetworkSet et DeviceTopology sont formellement définis dans l'IEC 62541-100.
 - DeviceTopology est utilisé pour agréger les Réseaux du niveau supérieur qui fournissent un accès à toutes les instances qui constituent la Topologie d'Appareil ((sous-)réseaux, appareils et éléments de communication). Certains éléments sont montrés en exemple ici et sont signalés par la couleur verte.
 - Tous les Appareils instanciés sont des composants de l'Objet DeviceSet, c'est-à-dire qu'ils existent dans l'AddressSpace indépendamment de la Topologie d'Appareil. Tous les réseaux sont des composants de l'Objet NetworkSet.

Les Serveurs FDI peuvent soit créer des Objets d'Appareil automatiquement, soit montrer uniquement les types disponibles (dossier SupportedTypes) et laisser l'utilisateur créer ses propres instances. Lors de l'utilisation d'une communication native, le système fournira typiquement la Topologie d'Appareil sans devoir la faire configurer par le Client FDI au préalable.

7 Modèle d'Appareil pour FDI

7.1 Généralités

Comme mentionné ci-dessus, l'IEC 62541-100 spécifie les types fondamentaux nécessaires pour le FDI comme le TopologyElementType, le DeviceType et le ProtocolType (le protocole de bus de terrain). L'Article 7 reprend brièvement les éléments de conception importants spécifiés dans l'IEC 62541-100 et spécifie des types supplémentaires qui ne sont pas présents dans l'IEC 62541-100.

La prise en charge de la Propriété DeviceHealth est obligatoire pour les Objets Appareil. Des informations détaillées sur le modèle et la définition formelle sont spécifiées dans l'IEC 62541-100.

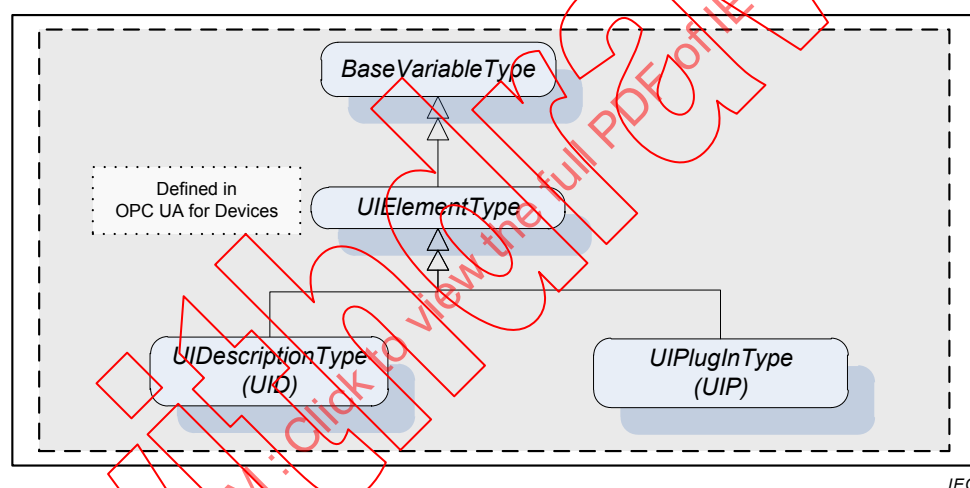
7.4 Eléments de l'interface utilisateur

7.4.1 Généralités

L'IEC 62541-100 définit, de manière abstraite, la façon dont les Serveurs peuvent exposer des éléments d'interface utilisateur aux Clients, afin d'afficher une interface utilisateur spécifique à un FunctionalGroups d'un TopologyElement.

Le paragraphe 7.4 spécifie deux types d'élément utilisateur concrets: les éléments d'interface utilisateur descriptive (les UID) et les éléments d'interface utilisateur programmée (exécutable) (les UIP). Les UIP ne sont jamais référencés directement à partir d'un FunctionalGroup. Ils sont toujours référencés indirectement à partir d'un UID au moyen de leur UipId.

La Figure 12 IEC 62541-100 présente la hiérarchie de types des éléments d'interface utilisateur définis dans l' et dans la présente norme.



Anglais	Français
Defined in OPC UA for Devices	Défini dans l'OPC UA pour les Appareils
UID (UI Description)	UID (Description d'Interface Utilisateur)
UIP (UI Plug-in)	UIP (Plug-in d'Interface Utilisateur)

Figure 12 – Hiérarchie des Types UserInterface

7.4.2 Type UI Description

Les Serveurs FDI peuvent fournir un élément d'interface utilisateur descriptive (un UID) pour chaque FunctionalGroup. Un tel élément sera fourni par le Client FDI. Le UIDescriptionType est formellement spécifié dans le Tableau 1.

Tableau 1 – Définition du UIDescriptionType

Attribut	Valeur				
BrowseName	UIDescriptionType				
DataType	String				
IsAbstract	False				
Références	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Hérite des Propriétés de l'UIElementType défini dans l'IEC 62541-100.					

L'attribut de Valeur fournit l'UID sous forme de Chaîne contenant un élément XML. Voir l'IEC 62769-2 pour la syntaxe des éléments UID. Le schéma XML pour la Valeur UID de tous les appareils exposés respecte la même Version de Technologie FDI que celle qui est indiquée par la Propriété FDIserverVersion (voir Article 14).

7.4.3 Type UI Plug-in

Un Plug-in d'Interface Utilisateur (UIP = User Interface Plug-in.) est un module logiciel qui est hébergé et exécuté par un Client FDI. UIP = User Interface Plug-in. Contrairement à une Description d'Interface Utilisateur (UID = User Interface Description.), il s'agit d'un élément d'UI exécutable.

Les détails sur l'hébergement et l'exécution des Plug-ins sont spécifiés dans l'IEC 62769-2. L'UIPlugInType est formellement spécifié dans le Tableau 2.

Tableau 2 – Définition du UIPlugInType

Attribut	Valeur				
BrowseName	UIPlugInType				
DataType	Byte				
ValueRank	1 – matrice unidimensionnelle				
ArrayDimensions	Uint32[N] – la longueur (nombre d'octets) de la matrice				
IsAbstract	False				
Références	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Hérite des Propriétés de l'UIElementType défini dans l'IEC 62541-100.					
HasProperty	Variable	UIVariantVersion	String	PropertyType	Obligatoire
HasProperty	Variable	FDITechnologyVersion	String	PropertyType	Obligatoire
HasProperty	Variable	RuntimeId	String	PropertyType	Obligatoire
HasProperty	Variable	CpuInformation	String	PropertyType	Obligatoire
HasProperty	Variable	PlatformId	String	PropertyType	Obligatoire
HasProperty	Variable	Style	String	PropertyType	Obligatoire
HasProperty	Variable	StartElementName	String	PropertyType	Obligatoire
HasComponent	Object	Documentation		FolderType	Facultatif

Les UIP peuvent exister dans de nombreuses variantes pour différentes plateformes ou pour supporter différentes versions. L'UiId (un identifiant unique défini dans le Paquetage FDI) identifie l'UIP, pas une variante spécifique.

Les UIP ne doivent pas être exposés dans l'AddressSpace. Avec l'UiId, le Client FDI peut récupérer les NodeId des Variantes UIP et leurs propriétés, en appelant le Service TranslateBrowsePathToNodeIds de l'OPC UA avec un NodeId d'Appareil en tant que StartingNode et la liste suivante de noms relatifs:

- "UIPSet/<UiId>"
- "UIPSet/<UiId>/RuntimeId"
- "UIPSet/<UiId>/CpuInformation"
- "UIPSet/<UiId>/PlatformId"
- "UIPSet/<UiId>/FDITechnologyVersion"

- "UIPSet/<UiId >/Style"
- "UIPSet/<UiId >/StartElementName"
- "UIPSet/<UiId >/UIPVariantVersion"

NOTE "UIPSet" est un identifiant du Serveur et n'est pas nécessairement un Nœud dans l'AddressSpace.

Le Serveur FDI renvoie des matrices de NodeIds pour chaque nom relatif. Le nombre d'entrées dans chaque matrice correspond au nombre de Variants UIP pour l'UiId. Le Client FDI peut lire les valeurs de propriété en utilisant les NodeId reçus et choisir la Variante UIP appropriée en fonction des FDITechnologyVersion, RuntimeId, CpuInformation et PlatformId.

L'Attribut Value fournit l'exécutable de l'UIP. La représentation exacte dépend de la technologie (voir l'IEC 62769-6). L'Attribut ArrayDimensions doit spécifier la taille (en nombre d'octets) de l'UIP.

Les Clients FDI ont besoin d'être capables de manipuler des UIP de grande taille. Une simple opération Read peut ne pas suffire pour lire des UIP de grande taille, en raison des limites configurées soit dans le Client FDI, soit dans la pile du Serveur FDI. La taille par défaut d'une matrice d'octets est de 1 Mégaoctet. Les Clients FDI peuvent utiliser l'IndexRange dans le Service Read de l'OPC UA (voir l'IEC 62541-4) pour lire une UIP en fragments de 1 Mo, par exemple. Il est laissé au soin du Client FDI de choisir entre démarrer sans indice et répéter avec un indexRange uniquement après une erreur ou de toujours utiliser un indexRange.

Les Propriétés suivantes aident le Client FDI à identifier l'UIP le plus adéquat pour son environnement:

- UIPVariantVersion:
La version de cette Variante UIP.
- FDITechnologyVersion:
Version de la technologie FDI en fonction de laquelle l'UIP est développé. Un UIP doit toujours être capable de fonctionner dans un système client/serveur avec la même version majeure et une version mineure/de maintenance différente.
- RuntimeId:
Environnement d'exécution de l'UIP tel qu'il est spécifié dans l'IEC 62769-6.
- CpuInformation:
Fournit les informations supplémentaires à propos de l'environnement d'exécution associé avec le RuntimeId. Les valeurs autorisées sont spécifiées dans l'IEC 62769-6.
- PlatformId définit le type de plateforme prenant en charge cette Variante d'UIP. Un Client FDI peut choisir une Variante d'UIP particulière si elle correspond à la plateforme du Client FDI (voir IEC 62769-4 pour les définitions concrètes).
 - "Station de travail" – avec les résolutions d'écran, les capacités de mémoire et les périphériques d'entrée disponibles (tels que souris et clavier) les plus courants
 - "Mobile" – résolution d'écran, mémoire et périphériques d'entrée possibles limités.
- Style:
Détermine si l'UIP doit s'exécuter de manière "modale" ou "amodale", tel que défini dans l'IEC 62541-4. Actuellement, les valeurs "Dialog" et "Window" sont définies. "Dialog" nécessite une fenêtre modale, tandis qu'un UIP au modèle "Window" sera invoqué dans une fenêtre modale ou amodale, tel que défini dans l'IEC 62769-2.
- StartElementName:
Élément nécessaire pour démarrer cette Variante d'UIP. L'IEC 62769-6 spécifie la façon dont ces informations sont utilisées lors de l'activation de l'UIP.

Les documents fournis pour une Variante d'UIP sont exposés sous la forme de Variables organisées dans un dossier Documentation. Dans la plupart des cas, ils représenteront un manuel de produit, qui peut exister sous la forme d'un ensemble de documents. Ces informations peuvent être récupérées par la lecture de la valeur Variable qui est représentée

sous la forme d'une ByteString. La ByteString complète doit être interprétée sous la forme d'un fichier PDF. Les Clients FDI ont besoin de savoir que les contenus représentés par ces variables peuvent être volumineux. Une simple opération Read peut ne pas suffire pour lire des valeurs de grande taille, en raison des limites configurées, soit dans le Client FDI, soit dans la pile du Serveur FDI. La taille par défaut d'une matrice d'octets est de 1 Mégaoctet. Il est recommandé que les Clients FDI utilisent l'IndexRange dans le Service Read de l'OPC UA (voir l'IEC 62541-4) pour lire ces Variables en plusieurs fragments, par exemple, en fragments de 1 Mo.

7.5 Informations de support spécifique au type

Chaque DeviceType peut comporter un ensemble de données complémentaires. Il s'agit principalement d'images, de documents ou de données spécifiques à un protocole. Les différents types d'informations sont organisés en différents dossiers.

Voir l'IEC 62541-100 pour la définition formelle des informations de support.

7.6 Actions

7.6.1 Vue d'ensemble

Les Actions sont des opérations exécutées dans le Serveur FDI au nom d'un élément de topologie. Une fois invoquées avec la Méthode InvokeAction, les Actions peuvent effectuer de nombreuses transitions d'état jusqu'à la fin de la procédure. L'état réel est accessible via une Variable transitoire et non consultable dont le NodeId est retourné par la Méthode InvokeAction.

Les Clients FDI peuvent s'abonner à cette Variable pour recevoir les mises à jour concernant l'exécution de l'Action (données d'Action). Les données d'Action peuvent rapporter une transition d'état ou une demande vers le Client FDI pour lequel l'entrée est nécessaire pour la continuation. Le Client FDI peut reprendre l'exécution avec la Méthode RespondAction et soumettre les données demandées.

La Figure 13 présente la manière dont les Actions sont intégrées dans un TopologyElement. Cela est défini formellement dans le Tableau 3.

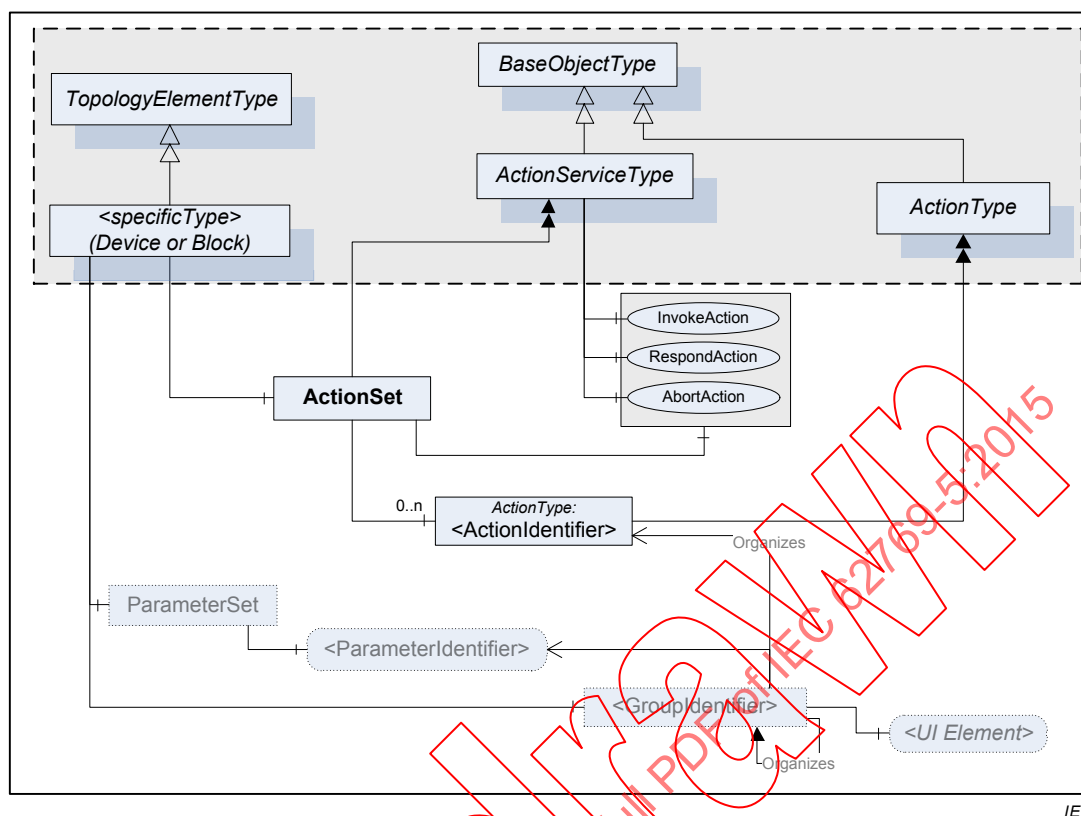


Figure 13 – Intégration des Actions au sein de TopologyElement

Tableau 3 – TopologyElementType avec suppléments pour les Actions

Attribut	Valeur				
BrowseName	TopologyElementType				
IsAbstract	True				
Références	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Hérite des Propriétés du BaseObjectType défini dans l'IEC 62541-5]					
HasComponent	Object	ParameterSet		BaseObjectType	Facultatif
HasComponent	Object	MethodSet		BaseObjectType	Facultatif
HasComponent	Object	<GroupIdentifier>		FunctionalGroupType	OptionalPlaceHolder
HasComponent	Object	Identification		FunctionalGroupType	Facultatif
HasComponent	Object	Lock		LockingServicesType	Facultatif
HasComponent	Object	ActionSet		ActionServiceType	Facultatif

ActionSet est utilisé pour agréger les Actions pour un TopologyElement spécifique. Il est ajouté au TopologyElementType afin de pouvoir également l'utiliser sur n'importe quel sous-type. Cet Objet est uniquement disponible dans une instance s'il existe des Actions pour ce TopologyElement.

Les extensions liées à l'Action pour l'ObjectType FunctionalGroup sont formellement définies dans le Tableau 4.

Tableau 4 – FunctionalGroupType avec suppléments pour les Actions

Attribut	Valeur				
BrowseName	FunctionalGroupType				
IsAbstract	False				
Références	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Hérite des Propriétés du FolderType défini dans l'IEC 62541-5]					
HasComponent	Object	<GroupIdentifier>		FunctionalGroupType	OptionalPlaceholder
Organizes	Variable	<ParameterIdentifier>		BaseDataVariableType	OptionalPlaceholder
Organizes	Method	<MethodIdentifier>			OptionalPlaceholder
HasComponent	Variable	UIElement	BaseDataType	UIElementType	Facultatif
Organizes	Object	<ActionIdentifier>		ActionType	Facultatif

<ActionIdentifier> se réfère aux Actions disponibles et exposées pour ce FunctionalGroup.

7.6.2 Type Action

Cet ObjectType définit la structure d'une Action. Cela est défini formellement dans le Tableau 5.

Tableau 5 – Définition du ActionType

Attribut	Valeur				
BrowseName	ActionType				
IsAbstract	True				
Références	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Le sous-type du BaseObjectType est défini dans l'IEC 62541-5.					

Le Client FDI détermine les arguments d'invocation requis pour une Action depuis l'UID.

7.6.3 Type ActionService

L'ActionServiceType définit les Méthodes pour invoquer et commander les Actions. Les instances de ce type agrègent les Actions pour un élément de topologie spécifique. L'ActionServiceType est formellement spécifié dans le Tableau 6. Son utilisation est présentée dans la Figure 13.

Tableau 6 – Définition du ActionServiceType

Attribut	Valeur				
BrowseName	ActionServiceType				
IsAbstract	False				
Références	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Le sous-type du BaseObjectType est défini dans l'IEC 62541-5.					
HasComponent	Method	InvokeAction			Obligatoire
HasComponent	Method	RespondAction			Obligatoire
HasComponent	Method	AbortAction			Obligatoire
HasComponent	Object	<ActionIdentifier>		ActionType	Facultatif

L'ActionServiceType et chaque instance de ce Type partagent les mêmes Méthodes. Le NodeId de ces Méthodes sera fixé et défini dans la présente norme. Les Clients FDI ne doivent donc pas naviguer pour chercher ces Méthodes. Ils peuvent utiliser le NodeId fixé comme MethodId pour le Service Call.

Le StatusCode de l'OPC UA, Bad_MethodInvalid, doit être retourné depuis le Service Call pour les éléments pour lesquels les Méthodes ActionService ne sont pas prises en charge.

<ActionIdentifier> désigne une ou plusieurs Actions. Les Actions (Objets de l'ActionType) existent uniquement dans des instances d'ActionServiceType, c'est-à-dire, lorsqu'une

instance de cet ObjectType est ajoutée à un TopologyElement. Il n'existe pas d'Actions (Objets de l'ActionType) dans l'ActionServiceType lui-même.

7.6.4 Objet ActionService

La prise en charge d'ActionService pour un Objet est déclarée en agrégeant une instance du Type ActionService conformément à la Figure 14.

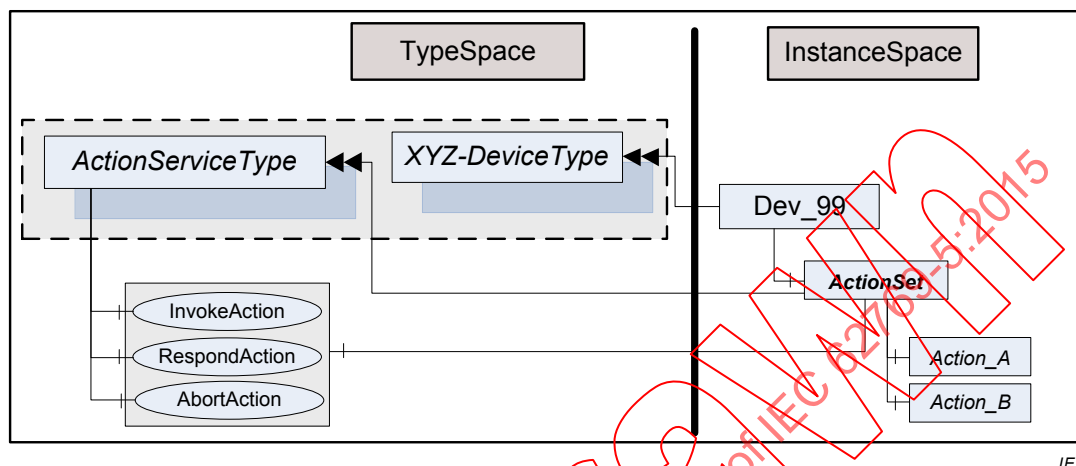


Figure 14 – Service Action

Cet Objet est utilisé en tant que conteneur pour les Méthodes ActionService et doit comporter le BrowseName ActionSet. Cela est défini formellement dans le Tableau 28. HasComponent est utilisé comme référence d'un TopologyElement (par exemple, un Appareil) vers son Objet "ActionService".

L'ActionServiceType et chaque Objet ActionSet partagent les mêmes Méthodes. Les Actions seront généralement partagées par toutes les instances d'un même Type d'Appareil.

7.6.5 Méthode InvokeAction

InvokeAction est utilisé pour lancer une Action. Il retourne immédiatement un résultat après la création du diagramme d'états. Le composant "ActionSet" du TopologyElement (Appareil) au nom duquel l'Action doit être invoquée est spécifié via l'argument ObjectId du Service Call.

Le verrouillage explicite est requis. Si l'Appareil n'a pas été verrouillé, le Serveur FDI rejettera la demande.

Après qu'InvokeAction retourne un résultat, le Client FDI doit s'abonner à l'Attribut Value de l'ActionNodeId. L'Attribut Value contient un élément XML (DataType = String (Chaîne)) qui reflète l'état actuel de l'Action, ainsi que des données supplémentaires (en fonction de l'état). Le Client FDI obtiendra alors une notification DataChange à chaque changement d'état de l'Action.

Voir l'IEC 62769-2 pour les diagrammes d'états de l'Action ainsi que pour le Schéma XML de la valeur ActionNodeId.

Les Serveurs FDI doivent mettre en mémoire-cache les données d'états de l'Action pendant une durée appropriée (quelques secondes), afin qu'aucune information d'états ne soit perdue avant que le Client FDI ait eu le temps de s'abonner.

La signature de cette Méthode est spécifiée ci-dessous. Le Tableau 7 et le Tableau 8 spécifient respectivement les arguments et la représentation de l'AddressSpace.

Signature

```

InvokeAction (
    [in] String      ActionName,
    [in] String      MethodArguments,
    [out] NodeId     ActionNodeId,
    [out] Int32      InvokeActionError);

```

Tableau 7 – Arguments de la Méthode InvokeAction

Argument	Description
ActionName	Portion String (chaîne) du BrowseName de l'Action qui est utilisée dans l'instance ActionServiceType.
MethodArguments	Document XML qui contient les arguments d'entrée de l'Action (le cas échéant). Le Schéma XML ListOfActionArguments défini dans l'IEC 62769-2 est utilisé pour le paramètre MethodArguments.
ActionNodeId	Nœud non consultable du type Variable. Ce nœud est utilisé à la fois pour identifier l'instance du diagramme d'état de l'Action et pour accéder aux informations d'états de l'Action.
InvokeActionError	0 – OK. -1 - E_LockRequired – l'élément n'est pas verrouillé comme requis -2 - E_UnknownAction – le nom transmis n'est pas une action valide pour cet élément.

Tableau 8 – Définition de l'AddressSpace de la Méthode InvokeAction

Attribut	Valeur				
BrowseName	InvokeAction				
Références	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Obligatoire
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Obligatoire

7.6.6 Méthode RespondAction

RespondAction est utilisé par le Client FDI pour fournir la valeur demandée ou la sélection d'une demande de fonction de l'UI (le cas échéant). Le composant "ActionSet" du TopologyElement au nom duquel l'Action a été invoquée est spécifié via l'argument ObjectId du Service Call (Appel).

La signature de cette Méthode est spécifiée ci-dessous. Le Tableau 9 et le Tableau 10 spécifient respectivement les arguments et la représentation de l'AddressSpace.

Signature

```

RespondAction (
    [in] NodeId      ActionNodeId,
    [in] String      Response,
    [out] Int32      RespondActionError);

```

Tableau 9 – Arguments de la Méthode RespondAction

Argument	Description
ActionNodeId	NodeId d'une Variable transitoire qui représente et identifie l'Action en cours d'exécution. Cet Id est retourné par la Méthode InvokeAction.
Response	Document XML qui contient la réponse (valeur ou sélection). Voir l'IEC 62769-2 pour le Schéma XML pour le paramètre Response.
RespondActionError	0 – OK -1 – E_InvalidAction – le NodeId ne fait pas référence à une action existante -2 – E_InvalidResponse – les données de réponse transmises n'ont pas pu être interprétées.

Tableau 10 – Définition de l'AddressSpace de la Méthode RespondAction

Attribut	Valeur				
BrowseName	RespondAction				
Références	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Obligatoire
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Obligatoire

7.6.7 Méthode AbortAction

AbortAction est utilisé par le Client FDI pour abandonner l'exécution d'une Action. L'appel de cette Méthode retourne immédiatement un résultat. Le Client FDI est informé via un événement notification sur la Variable ActionNodeId lorsque l'Action se met dans l'état Aborting (Abandon).

Le composant "ActionSet" du TopologyElement au nom duquel l'Action a été invoquée est spécifié via l'argument ObjectId du Service Call (Appel).

La signature de cette Méthode est spécifiée ci-dessous. Le Tableau 11 et le Tableau 12 spécifient respectivement les arguments et la représentation de l'AddressSpace.

Signature

```

AbortAction (
    [in]   NodeId           ActionNodeId,
    [out]  Int32            AbortActionError);
  
```

Tableau 11 – Arguments de la Méthode AbortAction

Argument	Description
ActionNodeId	NodeId d'une Variable transitoire qui représente et identifie l'Action en cours d'exécution. Cet Id est retourné par la Méthode InvokeAction.
AbortActionError	0 – OK -1 – E_InvalidAction – le NodeId ne fait pas référence à une action existante

Tableau 12 – Définition de l'AddressSpace de la Méthode AbortAction

Attribut	Valeur				
BrowseName	AbortAction				
Références	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Obligatoire
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Obligatoire

8 Réseau et connectivité

Les éléments d'information de réseau et de connexion sont requis pour créer les topologies de communication.

Un réseau représente les moyens de communication pour les Appareils auxquels il est connecté. Il comprend les technologies filaires ou sans fil. Les ConnectionPoints représentent l'interface (carte d'interface) d'un Appareil à un Réseau. Un sous-type spécifique doit être défini pour chaque protocole.

Ces éléments sont décrits et formellement définis dans l'IEC 62541-100.

9 Fonctions utilitaires

9.1 Vue d'ensemble

L'Article 9 fournit des services spécifiques pour certains éléments d'information de FDI.

9.2 Locking

Le Locking (verrouillage) est le moyen d'éviter des modifications simultanées sur un Appareil ou un Réseau et leurs composants. Les Clients FDI doivent utiliser les Services de Verrouillage pour tout changement (par exemple, les opérations d'écriture ou les invocations d'Actions).

L'objectif principal du verrouillage d'un Appareil est d'éviter les modifications simultanées d'un appareil. L'objectif principal du verrouillage d'un Réseau est de pouvoir éviter les modifications simultanées d'une topologie.

Lors du verrouillage d'un Appareil, le verrou s'applique toujours à la fois pour les versions en ligne et hors ligne.

Lors du verrouillage d'un Appareil Modulaire, le verrou s'applique à l'Appareil complet (y compris tous les modules). De la même manière, lors du verrouillage d'un Appareil orienté Bloc, le verrou s'applique à l'Appareil complet (y compris tous les blocs).

Si aucun verrou n'est appliqué à l'Appareil de niveau supérieur (pour Appareil Modulaire ou Appareil Bloc), les sous-appareils ou les blocs, respectivement, peuvent être verrouillés indépendamment.

Lors du verrouillage d'un Réseau, le verrou s'applique au Réseau et à tous les Appareils connectés. Si un des Appareils connectés fournit un accès à un Réseau subordonné (comme une passerelle), le Réseau subordonné et ses Appareils connectés sont également verrouillés.

Le LockingService est entièrement décrit et formellement défini dans l'IEC 62541-100.

9.3 EditContext

9.3.1 Vue d'ensemble

Un Editcontext peut être utilisé pour apporter des changements aux valeurs Variable visibles du Serveur sans les appliquer à l'Appareil. Le Serveur FDI fournit le concept EditContext pour supporter les Clients dans leur tâche d'édition.

L'EditContext est spécifié en IEC 62541-3. Ce qui suit concerne le Modèle d'Information OPC UA, y compris les Méthodes pour maintenir les instances EditContext.

EditContext est exposé sous la forme d'une capacité AddIn, comparable à la technologie d'interface présente dans certains langages de programmation. Le service Editcontext est modélisé sous la forme d'un ObjectType et des instances de ce type sont ajoutées à l'Appareil avec un BrowseName prédéfini. Des exemples d'AddIn supplémentaires sont définis dans l'IEC 62541-100.

Lors de la lecture ou de l'abonnement aux valeurs Variable enregistrées dans un EditContext, le StatusCode spécifique au FDI peut se produire (voir Article 11):)

- Good_Edited permet de distinguer les valeurs qui ont été éditées qui n'ont pas été écrites sur l'Appareil.

- Uncertain_DominantValueChanged indique que des valeurs dépendantes ne sont pas valides et seront recalculées après application de la valeur dominante à l'appareil. Hors ligne, la valeur dépendante doit également être écrite par le Client.
- Good_DependentValueChanged indique qu'une valeur dépendante a été modifiée, mais que la modification n'a pas été appliquée à l'appareil.

9.3.2 EditContext Type

EditContextType comprend les Méthodes d'EditContext (EditContextMethods). Cela est défini formellement dans le Tableau 13.

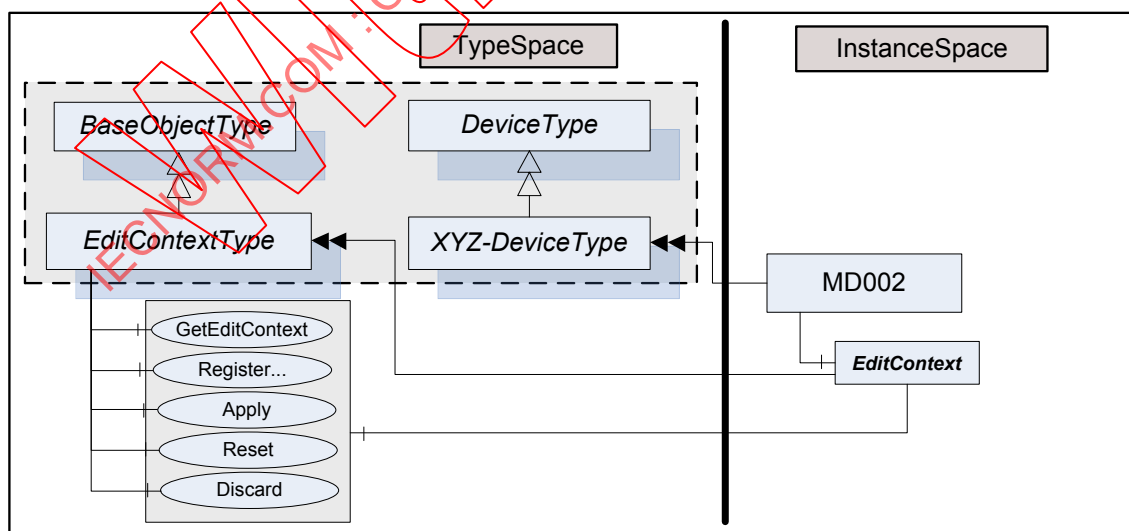
Tableau 13 – Définition du EditContextType

Attribut	Valeur				
BrowseName	EditContextType				
IsAbstract	False				
Références	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Le sous-type du BaseObjectType est défini dans l'IEC 62541-5.					
HasComponent	Method	GetEditContext			Obligatoire
HasComponent	Method	RegisterNodesById			Obligatoire
HasComponent	Method	RegisterNodesByRelativePath			Obligatoire
HasComponent	Method	Apply			Obligatoire
HasComponent	Method	Reset			Obligatoire
HasComponent	Method	Discard			Obligatoire

Le StatusCode Bad_MethodInvalid doit être retourné à partir du Service Call pour les éléments pour lesquels les Méthodes EditContext ne sont pas prises en charge. Bad_UserAccessDenied doit être retourné si l'Utilisateur Client n'a pas la permission d'appeler les Méthodes.

9.3.3 Objet EditContext

La prise en charge d'EditContext pour un Objet est déclarée en agrégeant une instance d'EditContextType conformément à la Figure 15.



IEC

Anglais	Français
Register...	Enregistrer...
Apply	Appliquer
Reset	Réinitialiser

Anglais	Français
Discard	Rejeter

Figure 15 – Type et instance EditContext

Cet Objet est utilisé en tant que conteneur pour les Méthodes EditContext et doit avoir le BrowseName EditContext. HasComponent est utilisé comme référence d'un Appareil vers son Objet "EditContext".

EditContextType et chaque instance peuvent partager les mêmes Méthodes.

9.3.4 Méthode GetEditContext

Retourne un EditContext comme spécifié dans l'IEC 62541-3.

La signature de cette Méthode est spécifiée ci-dessous. Le Tableau 14 et le Tableau 15 spécifient respectivement les arguments et la représentation de l'AddressSpace.

Signature

```
GetEditContext (
    [in] String      ParentId,
    [in] WindowMode TargetWindowMode,
    [out] String     EditContextId,
    [out] Int32      GetEditContextStatus);
```

Tableau 14 – Arguments de la Méthode GetEditContext

Argument	Description
ParentId	Si Null (vide), une instance racine est demandée. Dans le cas contraire, le Client utilise l'identifiant d'un EditContext acquis précédemment, et indique ainsi qu'il créera une sous-fenêtre.
TargetWindowMode	Une énumération qui indique l'élément d'Interface Utilisateur utilisé pour ce contexte. 1 – Fenêtre modale 2 – Fenêtre non modale 3 – UIP
EditContextId	Un identifiant de chaîne créé par le Serveur.
GetEditContextStatus	0 – OK -1 – E_NotSupported – l'élément ne supporte pas le Service EditContext

Tableau 15 – Définition de l'AddressSpace de la Méthode GetEditContext

Attribut	Valeur				
BrowseName	GetEditContext				
Références	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Obligatoire
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Obligatoire

9.3.5 Méthode RegisterNodes

Cette méthode est utilisée pour enregistrer des Nœuds avec un EditContext. Elle retourne de nouveaux NodeId qui doivent être utilisés pour adresser ce Nœud au sein de l'EditContext.

La signature de cette Méthode est spécifiée ci-dessous. Le Tableau 16 et le Tableau 17 spécifient respectivement les arguments et la représentation de l'AddressSpace.

Signatures

```
RegisterNodes (
    [in] String EditContextId,
    [in] RegistrationParameters[] NodesToRegister,
    [out] RegisterNodesResult RegisterNodesStatus);
```

Tableau 16 – Arguments de la Méthode RegisterNodes

Argument	Description
EditContextId	Identifiant d'un EditContext acquis précédemment avec un appel GetEditContext.
NodesToRegister	Une matrice de structures pour chaque nœud à enregistrer.
RegisterNodesStatus	Une structure comprenant l'ensemble des statuts de l'exécution et des données de résultat, pour chaque nœud à enregistrer.

Tableau 17 – Définition de l'AddressSpace de la Méthode RegisterNodes

Attribut	Valeur				
BrowseName	RegisterNodes				
Références	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Obligatoire
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Obligatoire

Le Type de Données RegistrationParameters définit un Nœud unique qui doit être enregistré. Ses éléments sont définis dans le Tableau 19.

Tableau 18 – Structure du type de données RegistrationParameters

Nom	Type	Description										
RegistrationParameters	structure	Cette structure spécifie un des nœuds devant être enregistrés.										
path	RelativePath	RelativePath pour ce nœud à enregistrer. Le type RelativePath est défini dans l'IEC 62541-4.										
selectionFlags	UInt32	Un masque de bits qui identifie les NodeId spécifiques à EditContext qui doivent être retournés dans la structure RegisterNodesResult. La valeur de ce paramètre doit contenir au moins une des valeurs suivantes. Aucune valeur ne sera rejetée avec E_InvalidSelectionFlags. <table><tr><th>Valeur de bit</th><th>NodeId à retourner</th></tr><tr><td>0x0000 0001</td><td>Online / ContextNodeId</td></tr><tr><td>0x0000 0002</td><td>Online / DeviceNodeId</td></tr><tr><td>0x0000 0004</td><td>Offline / ContextNodeId</td></tr><tr><td>0x0000 0008</td><td>Offline / DeviceNodeId</td></tr></table>	Valeur de bit	NodeId à retourner	0x0000 0001	Online / ContextNodeId	0x0000 0002	Online / DeviceNodeId	0x0000 0004	Offline / ContextNodeId	0x0000 0008	Offline / DeviceNodeId
Valeur de bit	NodeId à retourner											
0x0000 0001	Online / ContextNodeId											
0x0000 0002	Online / DeviceNodeId											
0x0000 0004	Offline / ContextNodeId											
0x0000 0008	Offline / DeviceNodeId											

Le type de données RegisterNodesResult inclut toutes les informations de statut et les données de résultat pour chaque Nœud qui doit être enregistré. Ses éléments sont définis dans le Tableau 19.

Tableau 19 – Structure du type de données RegisterNodesResult

Nom	Type	Description
RegisterNodesResult	structure	Cette structure spécifie le résultat de l'enregistrement
status	Int32	0 – OK – le champ registeredNodes contient un résultat pour chaque Nœud à enregistrer. -1 – E_InvalidId – l'EditContext spécifié est inconnu le champ registeredNodes est vide
registeredNodes	RegisteredNode[]	La liste contient les NodeIds spécifiques à EditContext pour le Nœud enregistré. Le Client doit utiliser ces NodeId pour tous les appels ultérieurs de Service OPC UA.
nodeStatus	Int32	0 – OK -1 – E_InvalidNode – Un Nœud invalide a été enregistré. Voir l'IEC 62541-3 pour plus de détails -2 – E_InvalidSelectionFlags – les RegistrationParameters pour ce Nœud ne contenaient pas de valeur
onlineContextNodeId	NodeId	Ce NodeId doit être utilisé pour adresser la représentation en ligne du Nœud dans EditContext. Voir l'IEC 62541-3 pour plus de détails
onlineDeviceNodeId	NodeId	Ce NodeId doit être utilisé pour adresser la représentation en ligne du Nœud dans l'Appareil. Voir l'IEC 62541-3 pour plus de détails
offlineContextNodeId	NodeId	Ce NodeId doit être utilisé pour adresser la représentation en ligne du Nœud dans EditContext. Voir l'IEC 62541-3 pour plus de détails
offlineDeviceNodeId	NodeId	Ce NodeId doit être utilisé pour adresser la représentation en ligne du Nœud dans l'Appareil. Voir l'IEC 62541-3 pour plus de détails

9.3.6 Méthode Apply

Cette Méthode est utilisée pour appliquer les valeurs qui ont été modifiées dans EditContext.

La signature de cette Méthode est spécifiée ci-dessous. Le Tableau 20 et le Tableau 21 spécifient respectivement les arguments et la représentation de l'AddressSpace.

Signature

```

Apply (
    [in] String      EditContextId,
    [out] ApplyResult ApplyStatus);

```

Tableau 20 – Arguments de la Méthode Apply

Argument	Description
EditContextId	Identifiant d'un EditContext acquis précédemment avec un appel GetEditContext.
ApplyStatus	Une structure comprenant l'ensemble des statuts de l'exécution et des informations de statut, pour chaque Variable transférée.

Tableau 21 – Définition de l'AddressSpace de la Méthode Apply

Attribut	Valeur				
BrowseName	Apply				
Références	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Obligatoire
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Obligatoire

Le DataType ApplyResult inclut toutes les informations d'état d'Apply et les informations d'état pour chaque Variable qui n'a pas pu être transmise. Ses éléments sont définis dans le Tableau 22 et le Tableau 19.

Tableau 22 – Structure du DataType ApplyResult

Nom	Type	Description
ApplyResult	structure	Cette structure est retournée en cas d'erreurs. Aucune donnée de résultat n'est retournée. Il est impossible d'effectuer d'autres appels avec le même TransferId.
status	Int32	0 – OK – le champ transferIncidents peut inclure des Variables individuelles ayant échoué. -1 – E_InvalidId – l'EditContext spécifié est inconnu
transferIncidents	TransferIncident[]	Si le service retourne normalement un résultat et la liste des TransferIncident est vide, tous les changements ont été appliqués et les valeurs éditées sont effacées. Dans le cas contraire, la liste contient des Variables qui n'ont pas pu être transmises correctement. Les Valeurs éditées sont préservées.
contextNodeId	NodeId	Le NodeId retourné depuis RegisterNodesById ou RegisterNodesByRelativePath.
statusCode	StatusCode	Le StatusCode de l'OPC UA est défini dans l'IEC 62541-4 et dans l'IEC 62541-8.
diagnostics	DiagnosticInfo	Informations de Diagnostic. Ce paramètre est vide si les informations de diagnostic n'ont pas été demandées dans l'en-tête de la demande, ou si aucune information de diagnostic n'a été rencontrée dans l'exécution de la demande. Le type DiagnosticInfo est défini dans l'IEC 62541-4.

9.3.7 Méthode Reset

Efface toutes les valeurs modifiées dans EditContext.

La signature de cette Méthode est spécifiée ci-dessous. Le Tableau 23 et le Tableau 24 spécifient respectivement les arguments et la représentation de l'AddressSpace.

Signature

```

NodeId
[in] String
[out] Int32

```

```

EditContextId,
ResetStatus);

```

Tableau 23 – Arguments de la Méthode Reset

Argument	Description
EditContextId	Identifiant d'un EditContext acquis précédemment avec un appel GetEditContext.
ResetStatus	0 – OK -1 – E_InvalidId – l'EditContext spécifié est inconnu

Tableau 24 – Définition de l'AddressSpace de la Méthode Reset

Attribut		Valeur			
BrowseName	Reset				
Références	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Obligatoire
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Obligatoire

9.3.8 Méthode Discard

Cette Méthode libère EditContext. Toutes les valeurs modifiées n'ayant pas été appliquées sont perdues.

La signature de cette Méthode est spécifiée ci-dessous. Le Tableau 25 et le Tableau 26 spécifient respectivement les arguments et la représentation de l'AddressSpace.

Signature

```

Discard(
[in] String      EditContextId,
[out] Int32     DiscardStatus);

```

Tableau 25 – Arguments de la Méthode Discard

Argument	Description
EditContextId	Identifiant d'un EditContext acquis précédemment avec un appel GetEditContext.
DiscardStatus	0 – OK -1 – E_InvalidId – l'EditContext spécifié est inconnu -2 – E_ChildExists – l'EditContext spécifié ne peut pas être libéré, car il existe une instance enfant.

Tableau 26 – Définition de l'AddressSpace de la Méthode Discard

Attribut	Valeur				
BrowseName	Discard				
Références	NodeClass	BrowseName	Data Type	TypeDefinition	ModellingRule
HasProperty	Variable	InputArguments	Argument[]	PropertyType	Obligatoire
HasProperty	Variable	OutputArguments	Argument[]	PropertyType	Obligatoire

9.4 Direct Device Access

9.4.1 Généralités

DirectDeviceAccess fournit les moyens pour communiquer avec un appareil dans la Topologie d'Appareil. Il sera utilisé par les UIP pour les opérations qui ne peuvent, ou du moins pas de manière aisée, être effectuées à travers l'accès du Modèle d'Information. Les cas d'utilisation comprennent la transmission de paquets de données de grande taille depuis ou vers l'appareil, par exemple, des données historiques ou un microprogramme ("firmware"). Il est généralement admis que les données auxquelles on accède directement ne seront pas reflétées dans le Modèle d'Information. DirectDeviceAccess ne doit pas influencer l'intégrité de la structure et des données du Modèle d'Information. Les Serveurs FDI peuvent être restrictifs en ce qui concerne le moment où ils autorisent l'utilisation de ces Méthodes.

Un Objet DirectDeviceAccess doit exister pour chaque Appareil pour lequel le Serveur FDI autorise l'accès direct via un UIP.

Le comportement suivant s'applique lors de l'utilisation de DirectDeviceAccess:

- Un seul Client FDI peut utiliser DirectDeviceAccess à un instant donné.
- DirectDeviceAccess nécessite un verrou. Si l'Appareil n'a pas été verrouillé, la demande sera rejetée.
- À cause du verrou, aucune opération d'écriture et aucune invocation de Méthode, provenant d'autres Clients FDI, ne sont autorisées pendant l'utilisation de DirectDeviceAccess. Cela comprend l'exécution des Actions.
- Si les valeurs d'Attribut des Paramètres peuvent avoir été changées du fait de DirectDeviceAccess, l'UIP doit mettre à True l'InvalidateCache dans l'argument EndDirectAccess.

9.4.2 Type DirectDeviceAccess

Le DirectDeviceAccessType fournit les Méthodes nécessaires pour ouvrir et fermer une connexion et pour transférer des données. La Figure 16 présente la définition du DirectDeviceAccessType. Cela est défini formellement dans le Tableau 27.

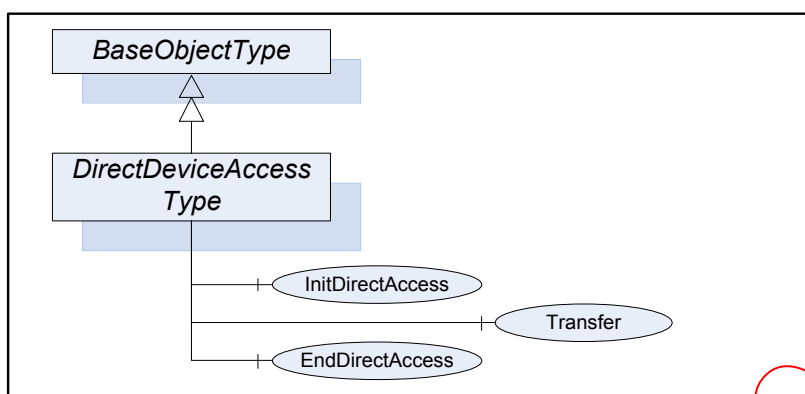


Figure 16 – DirectDeviceAccessType

Tableau 27 – Définition du DirectDeviceAccessType

Attribut	Valeur				
BrowseName	DirectDeviceAccessType				
IsAbstract	False				
Références	NodeClass	BrowseName	DataType	TypeDefinition	ModellingRule
Le sous-type du BaseObjectType est défini dans l'IEC 62541-5.					
HasComponent	Method	InitDirectAccess			Obligatoire
HasComponent	Method	Transfer			Obligatoire
HasComponent	Method	EndDirectAccess			Obligatoire

Le DirectDeviceAccessType et chaque instance de ce Type partagent les mêmes Méthodes. Le NodeId de ces Méthodes sera fixé et défini dans la présente norme. Les Clients FDI ne doivent donc pas naviguer pour chercher ces Méthodes. Ils peuvent utiliser le NodeId fixé comme MethodId pour le Service Call.

Le StatusCode de l'OPC UA, Bad_MethodInvalid, doit être retourné depuis le Service Call pour les éléments pour lesquels les Méthodes DirectDeviceAccess ne sont pas prises en charge.

9.4.3 Objet DirectDeviceAccess

La prise en charge de DirectDeviceAccess pour un Objet est déclarée en agrégeant une instance du Type DirectDeviceAccess conformément à la Figure 17.

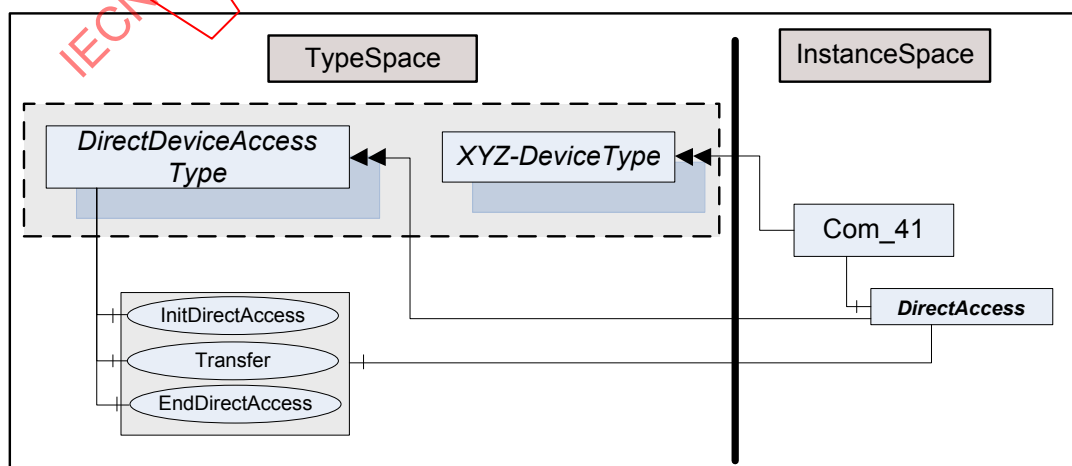


Figure 17 – Instance DirectDeviceAccess