IEC 62769-3

Edition 2.0 2021-02
REDLINE VERSION

INTERNATIONAL STANDARD

colour inside

Field device integration (FDI) – Part 3: ~~FDI~~ Server

IEC 62769-3:2021-02 RLV(en)

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

Tel.: +41 22 919 02 11
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 corrigendum or an amendment might have been published.

IEC publications search - webstore.iec.ch/advsearchform
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 It also gives information on projects, replaced and withdrawn publications.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 once a month by email.

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: sales@iec.ch.

IEC online collection - oc.iec.ch
Discover our powerful search engine and read freely all the publications previews. With a subscription you will always have access to up to date content tailored to your needs.

Electropedia - www.electropedia.org
The world's leading online dictionary on electrotechnology, containing more than 22 000 terminological entries in English and French, with equivalent terms in 18 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC 62769-3

Edition 2.0 2021-02
REDLINE VERSION

INTERNATIONAL STANDARD

colour inside

Field device integration (FDI) – Part 3: ~~FDI~~ Server

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

ICS 25.040.40; 35.100.05

ISBN 978-2-8322-9389-8

Warning! Make sure that you obtained this publication from an authorized distributor.

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

FIELD DEVICE INTEGRATION (FDI) –

Part 3: ~~FDI~~ Server

FOREWORD

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

This redline version of the official IEC Standard allows the user to identify the changes made to the previous edition IEC 62769-3:2015. A vertical bar appears in the margin wherever a change has been made. Additions are in green text, deletions are in strikethrough red text.

International Standard IEC 62769-3 has been prepared by subcommittee 65E: Devices and integration in enterprise systems, of IEC technical committee 65: Industrial-process measurement, control and automation.

This second edition cancels and replaces the first edition published in 2015. This edition constitutes a technical revision.

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

a) modification of the edit context concept to harmonize the IEC 61804 and the IEC 62769 series.

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

FDIS	Report on voting
65E/760/FDIS	65E/770/RVD

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

This document 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 document will remain unchanged until the stability date indicated on the IEC website under "http://webstore.iec.ch" in the data related to the specific document. At this date, the document 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 Rytoft~~
~~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, Marshaltown, 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.~~

The IEC 62769 series has the general title *Field Device Integration (FDI)* and the following parts:

– Part 1: Overview
– Part 2: FDI Client
– Part 3: FDI Server
– Part 4: FDI Packages
– Part 5: FDI Information Model
– Part 6: FDI Technology Mapping
– Part 7: FDI Communication Devices

- Part 100: Profiles – Generic Protocol Extensions
- Part 101-1: Profiles – Foundation Fieldbus H1
- Part 101-2: Profiles – Foundation Fieldbus HSE
- Part 103-1: Profiles – PROFIBUS
- Part 103-4: Profiles – PROFINET
- Part 109-1: Profiles – HART and WirelessHART
- Part 115-2: Profiles – Protocol-specific Definitions for Modbus RTU
- Part 150-1: Profiles – ISA 100.11a

FIELD DEVICE INTEGRATION (FDI) –

Part 3: ~~FDI~~ Server

1 Scope

This part of IEC 62769 specifies the FDI Server. 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 figure.

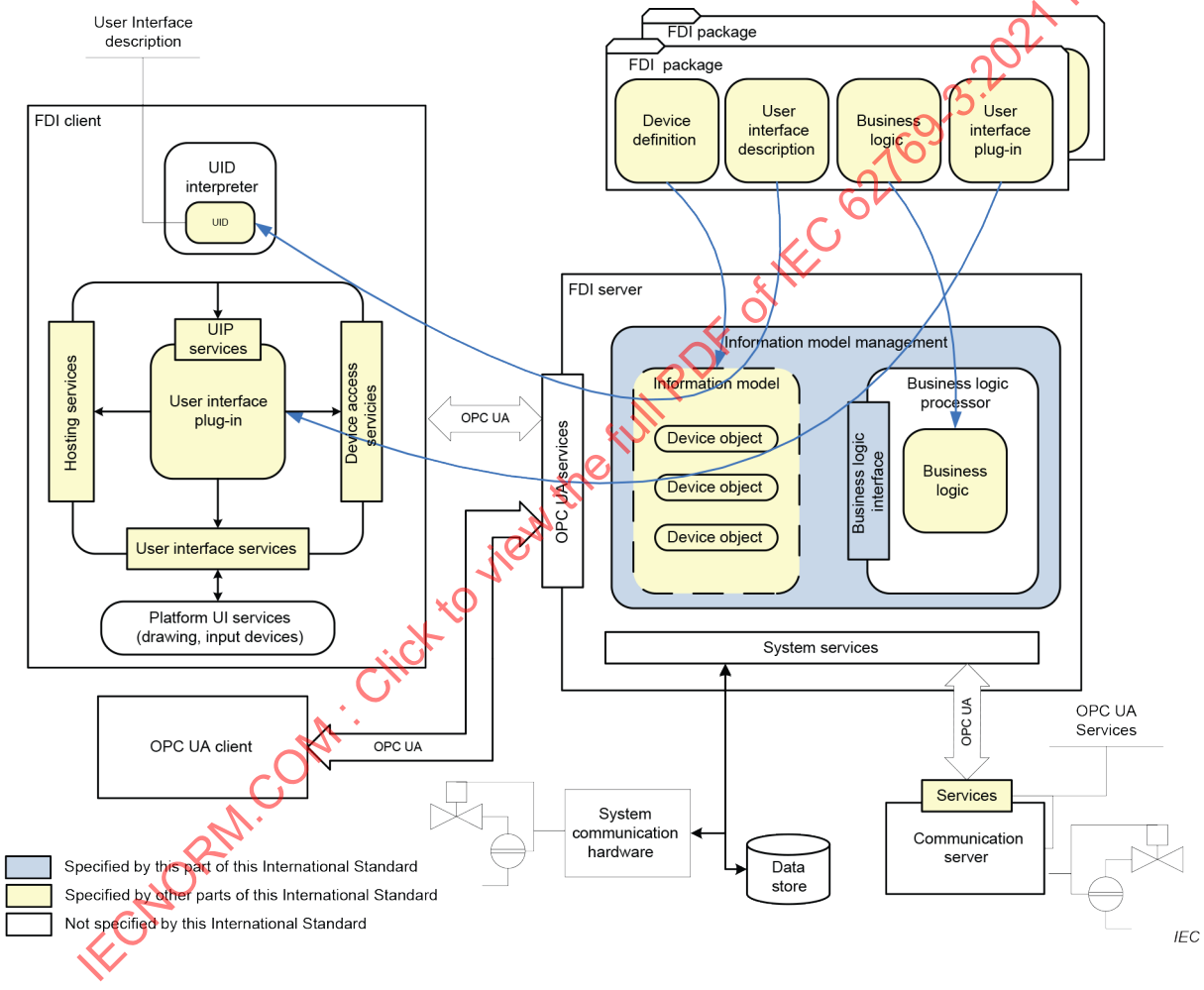

Figure 1 – FDI architecture diagram

2 Normative references

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

IEC 61804 (all parts), *Function blocks (FB) for process control and electronic device description language (EDDL)*

~~IEC 61804-3[1], *Function block (FB) for process control and Electronic Device Description Language (EDDL) – Part 3: EDDL syntax and semantics*~~

~~IEC 61804-4[2], *Function blocks (FB) for process control and Electronic Device Description Language (EDDL) – Part 4: EDD interpretation*~~

IEC 61804-4:2020, *Function blocks (FB) for process control and electronic device description language (EDDL) – Part 4: EDD interpretation*

~~IEC 62541 (all parts), *OPC unified architecture*~~

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

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

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 62769-5, *Field Device Integration (FDI) – Part 5: FDI Information Model*

~~NOTE IEC 62769-5 is technically identical to FDI-2025.~~

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 and ~~acronyms~~ conventions

3.1 Terms and definitions

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

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

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

3.1.1
Actions Proxy
internal FDI Server entity that encapsulates all the EDD Methods specified in an EDD Action definition

~~[1] To be published.~~

~~[2] To be published.~~

3.2 Abbreviated terms ~~and acronyms~~

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

3.3 Conventions

For the purposes of this document, the conventions given in IEC 62769-1 apply.

4 Overview

The structure for an FDI Server is shown in Figure 1.

FDI Servers that support connectivity with third-party FDI Clients shall support OPC UA. A vendor can provide both an FDI Server and one or more FDI Clients. In this case, the FDI Clients can communicate with the FDI Server through proprietary protocols.

An FDI Server communicates with devices via Native Communication (see 7.2.1) and/or Communication Devices (see IEC 62769-7).

An FDI Server provides information to FDI Clients through an Information Model (see IEC 62769-5) as follows.

- The Information Model includes information about Device Types and Device Instances. The information for a Device Instance includes offline data (engineering data), as well as online data (values from the physical device).
- The Information Model is created using information from FDI Packages. However, not all of the information in an FDI Package is reflected in the Information Model.
- Referential integrity of the Information Model is maintained using information from FDI Packages.
- FDI Packages can contain Attachments that contain device manuals and protocol-specific information (see IEC 62769-4). Those Attachments, including device manuals and protocol-specific support files, are exposed via the Information Model.
- FDI Device Packages contain information about device types (see IEC 62769-4). Each device type defined in a package is mapped to a distinct DeviceType node in the Information Model.
- FDI Profile Packages are used to provide interaction with devices for which an FDI Device Package does not exist (see IEC 62769-4).
- Multiple revisions of an FDI Package generate distinct DeviceType nodes in the Information Model (see IEC 62769-4).

FDI Packages contain digital signatures that allow an FDI Server to authenticate their contents (see IEC 62769-4). An FDI Server shall not use an FDI Package if the digital signature provided by the FDI Package is invalid.

An FDI Server shall verify the FDI Technology Version (see IEC 62769-1) of any FDI Package it uses to ensure the FDI Package is compatible with the FDI Server.

The resulting functional structure of an FDI server is described in Annex A.

5 Information Model

5.1 General

The FDI Server shall use the Device Definition of an FDI Package to maintain the Information Model.

The Device Definition can contain conditional expressions. Conditional expressions are used when a certain aspect of the Device Definition is not static but rather is dependent on the state of the device. Whenever the online or offline values of a Device Instance are modified, the FDI Server shall re-evaluate the relevant conditional expressions and modify the Information Model accordingly.

The evaluation of conditional expressions can invalidate variables in the Information Model. The FDI Server shall change the AccessLevel attribute of invalidated variables such that they are neither readable nor writable and the status of these variables shall be set to bad. Read and write service requests for invalidated variables shall return a failure.

The Device Definition can specify relationships between variables in a device. These relationships can impact the value of variables in the Information Model.

The FDI Server shall generate DataChange Notifications to any FDI Clients that are subscribing to Information Model elements that have changed.

FDI Packages provide Business Logic that is used by the FDI Server to maintain the integrity of the Information Model. The Business Logic specified in an FDI Package can invoke builtin functions that shall be implemented by the FDI Server. The builtin functions that shall be implemented by the FDI Server are specified in IEC 61804-5.

5.2 Online/Offline

5.2.1 Overview

The Information Model maintained by the FDI Server contains online and offline values. The online values reflect values in a physical component/device. The offline values reflect values stored in a configuration database.

The offline values are updated through write service requests from an FDI Client or Business Logic executed by the FDI Server. The offline values are not updated when the FDI Server reads data from the device or writes data to the device.

The online values in the Information Model are not updated through write service requests. Successful write service requests through the Information Model result in value changes in the physical devices. The online values in the Information Model will then be updated as a result of read service requests or subscriptions.

FDI Servers can provide a server-specific mechanism for creating Device Instances without the presence of physical hardware. The FDI Server creates these instances using information in FDI Packages. All read/write requests for online values for Device Instances with no physical device shall return an error.

The transfer of information between the offline values and the physical device is supported through the TransferToDevice and TransferFromDevice methods in the Information Model. These Methods shall implement the download and upload procedures, respectively, as specified in IEC 61804-4. When no implementation is provided based on IEC 61804-4, then these Methods shall return Bad_NotSupported, as per IEC 62541-4.

The Device shall have been locked prior to invoking these methods, as specified in IEC 62769-5.

5.2.2 Transfer to device

The TransferToDevice method shall implement the download procedure as specified in IEC 61804-4. This transfers the offline values to the physical device.

As a general rule, the FDI Server should not change the Online variable node when writing a value to the device. The Online variable node should be updated only in the process of read operations or subscriptions. Notwithstanding, as specified in IEC 62769-5, the FDI Server will reset any cached Value for the target Nodes in the Information Model so that they will be re-read next time they are requested.

The status information returned for each variable included in the write service request is used to compose the TransferResult, as specified in IEC 62769-5.

5.2.3 Transfer from device

The TransferFromDevice method shall implement the upload procedure as specified in IEC 61804-4. This transfers the values from the physical device to the offline values.

If any read operations from the device fail during upload, the corresponding offline value shall not be modified.

The status information returned for each variable included in the read service request is used to compose the TransferResult, as specified in IEC 62769-5.

5.3 Access privileges

Systems implement security and access policies based on a number of characteristics such as user role and plant area. FDI Servers use these policies, along with information in FDI Packages, to determine the access privileges granted to the user.

The elements of an FDI Package can be associated with one or more usage attributes. The FDI Server uses these attributes to set the UserAccessLevel attribute of Variables and the UserExecutable attribute of Methods. The usage attributes in an FDI Package are simply hints to be used by the FDI Server, i.e., they may be disregarded or overridden by the FDI Server. See also Annex B.

5.4 Private Parameters

The Parameters and Actions specified in an FDI Package may be declared private. Private Parameters and Actions shall not be browsable; they shall only be accessible through references from other elements of an FDI Package.

More specifically, the FDI Server shall support private Parameters and Actions as follows.

- The FDI Server shall create nodes in the Information Model for the private Parameters and Actions.
- The FDI Server shall not include information about private Parameters and Actions in a response to a Browse, BrowseNext, QueryFirst, or QueryNext service request.
- The FDI Server shall return the NodeIds of private Parameters and Actions when the name of a private Parameter or Action is passed to TranslateBrowsePathsToNodeIds.
- The FDI Server shall process a read/write service request for a private Parameter in the same way as it does for public (browsable) Parameters (see 5.7 and 5.8).

- The FDI Server shall execute private Actions in the same way as it does public (browsable) Actions (see 5.12).

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.

5.5 Locking

The FDI Server provides locking services to grant FDI Clients exclusive access to Device and Network elements in the Information Model. The locking services consist of a set of Methods and status information. The methods, and their behavior, are specified in IEC 62769-5.

The following behavior shall be implemented by the FDI Server to support locks.

- Locking applies to both online and offline nodes.
- Once locked by one FDI Client, any attempt to write to a Parameter or to execute an Action by another FDI Client shall be rejected.
- Locking is not required for read services.
- Parameters that are locked by one FDI Client can still be read by other FDI Clients, i.e., read requests on a Parameter that is locked are not rejected.

Internal use of the locking mechanism for maintaining the Information Model integrity is FDI Server vendor-specific.

Figure 2 illustrates a locking sequence with multiple service invocations during the locked state.

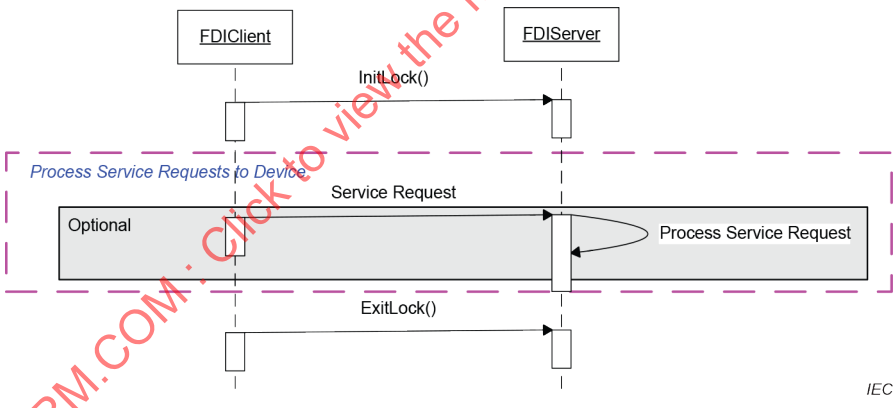

Figure 2 – Locking services

A service request that requires locking shall fail either partially or completely if no lock has been acquired by the FDI Client via InitLock prior to requesting the service. The FDI Client has to release the lock via ExitLock after all service requests have been completed.

~~NOTE~~ A write operation will partially fail, i.e., it will return a status code for each variable in the set of variables to be written since some may belong to devices that are locked and some to devices that are not locked.

FDI Servers may queue InitLock requests until a service for which a lock has been created completes and the lock has been released. However, such an optimization is not part of the standard behavior required of an FDI Server.

5.6 EditContext

5.6.1 Concept and usage model

The FDI Server provides the EditContext model to interact with Clients during their editing task. The concept is closely related to UIDs and fulfills the needs for Server-driven UI dialogs based on EDDL rules.

An EditContext can be used to make changes to Variable Values visible to the Server without applying them to the online or offline representation of a Device. The Server will apply business logic associated to the edited Variable, which – in some cases – causes changes to other Variable Values (e.g. if an engineering unit is changed) or the UID (e.g. a Variable becomes invisible). Thus, the Client can use an EditContext to modify (edit) Parameters such as engineering units, ranges and more, verify any side effects, and re-adjust the settings before applying the changes.

An FDI Server may implement different EditContext strategies:

- A single EditContext instance for all dialogues of an FDI Client.
- Multiple EditContext instances.
- Hierarchical EditContext instances.

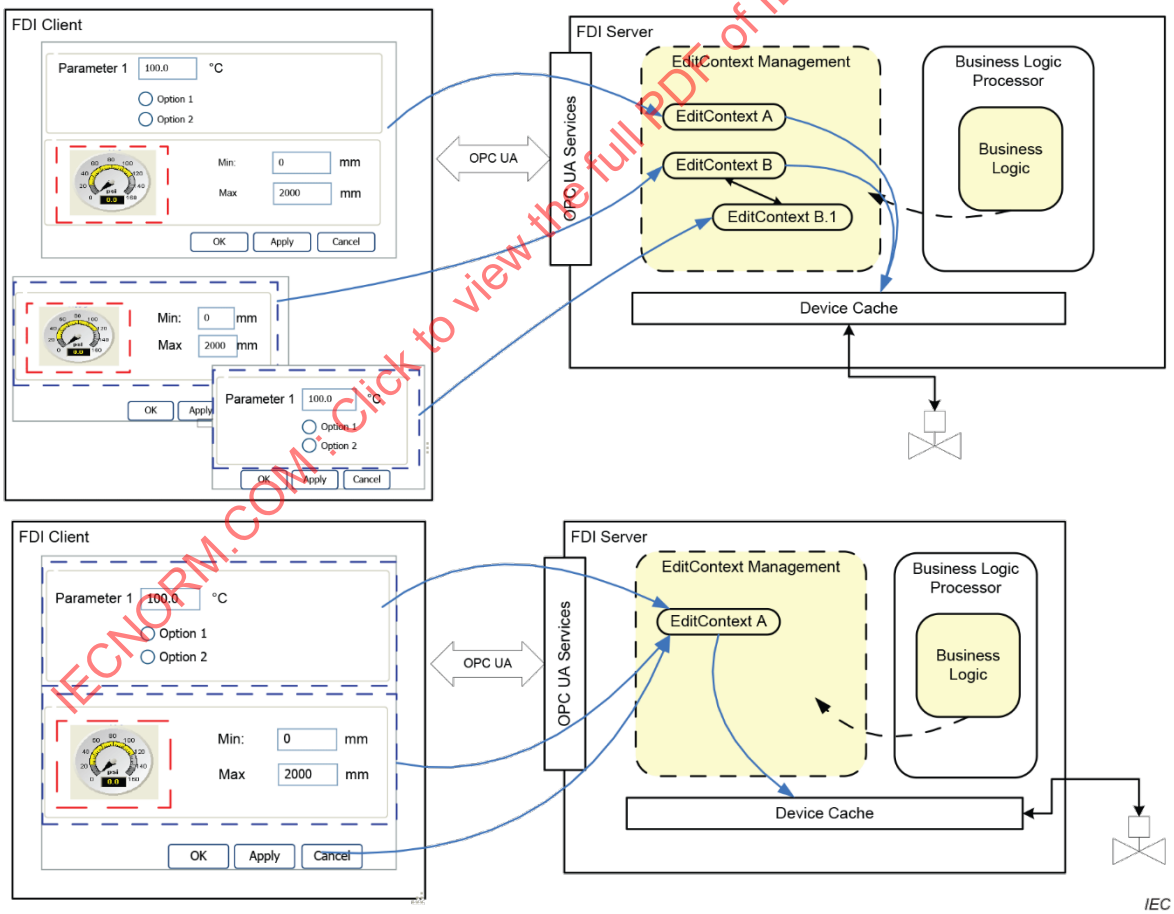

Figure 3 – EditContext models

Figure 3 shows two possible Server strategies and how the Client can adapt. In the lower scenario, the Server provides a single EditContext instance for all dialogs. Here, the Client

groups all dialogs and exposes a single set of buttons to Apply and Cancel, because it always concerns all edits.

In the upper scenario, the Server provides multiple EditContext instances, one of them as child of another one. Each instance can be addressed separately. If the changes in a child instance are applied, they are transferred to the parent. If the changes in a root instance are applied, they are transferred to the Device.

~~Parent-child dependencies:~~

- ~~A change for a Variable in the parent overwrites an edited Value for the same Variable in the Client.~~
- ~~The parent cannot be discarded before the child is discarded but the edited Values in the parent instance can be applied or reset.~~

The parent-child dependencies are defined in IEC 61804-4:2020, Clause 8.

5.6.2 Services

A set of Services is provided to the FDI Client to maintain EditContext instances (see IEC 62769-5 for a detailed description of these Services):

- GetContext – This Service is used to request an EditContext instance. The Client specifies certain characteristics for the Server to decide which EditContext instance to return. Depending on its internal strategy, the Server returns the same instance or new instances.
- RegisterNodes – The FDI Client has to register all Nodes of the Information Model that shall be maintained in an EditContext. It is possible to register Nodes of the online and of the offline representation of a Device. The result is new NodeIds that the Client shall use when calling Services to read, write, subscribe to Variables or to invoke Actions.
- Apply – Transfer the modified (edited) Variable Values to the parent (either a parent EditContext instance or the Device). If the same Variable has been edited in the parent instance it will be overwritten with a call of the Apply Service for the child.
- Reset – Clears all modifications. A Reset of already applied modifications is not possible.
- Discard – Deletes an EditContext instance (and its children). Edited Values that have not been applied are discarded. Once deleted, all registered NodeIds will be invalid. If such NodeIds are still subscribed, the Client is notified with proper StatusCodes.

The Client first calls GetEditContext to acquire an EditContext instance. It will then register the Nodes it wants to be part of it. The registration returns new NodeIds which can then be used for reading, writing or subscribing Variables and for calling Methods.

The Client can call GetEditContext multiple times, for instance when it opens an additional edit window or for a completely separate dialog (diagnosis in parallel to configuration). It is up to the Server strategy whether it returns a new instance or the same instance. The Client is expected to adapt its user interface to the EditContext strategy of the Server. See Figure 3 for how Clients may position the Apply and Cancel buttons so that the User clearly understands which changes ~~s/he applies or discards~~ are applied or discarded.

5.6.3 NodeIds

RegisterNode returns two NodeIds for each registered Node: a ContextNodeId and a DeviceNodeId. The Client uses these NodeIds when calling OPC UA Services to read, write and subscribe or call a Method.

Using the ContextNodeId addresses the Value in the EditContext instance. Using the DeviceNodeId addresses the Value in the Device.

5.6.4 Reading

Reading or subscribing a Variable with the ContextNodeId will return the edited Value from the EditContext instance. If no edited Value exists, the Value from the parent instance or the Device (online or offline) will be returned.

The StatusCode indicates whether the Value originates from the Device (StatusCode Good defined in IEC 62541) or from an EditContext instance (StatusCode Good_Edited defined in IEC 62769-5).

Reading or subscribing a Variable with the DeviceNodeId will return the Value from the Device (online or offline).

5.6.5 Writing

Writing to a Variable with the ContextNodeId modifies the Value in the EditContext instance.

Writing to a Variable with the DeviceNodeId modifies the Value in the Device (online or offline). Any edited Values for this Variable in the addressed EditContext instance or its parents will be reset.

5.6.6 Writing dominant and dependent Variables

~~Dependencies between dominant and dependent Variables are only evaluated when writing to the online Device, not in the EditContext. The following rules support FDI Clients when working with such Variables in the EditContext:~~

~~a) Change to the dominant Variable:~~

~~1) A change is made in Online EditContext~~

~~• The Client monitoring Variables in the EditContext will get~~

~~– Good_Edited for dominant Variable~~

~~– Uncertain_DominantValueChanged for dependent Variable.~~

~~• If dependent and dominant Variables have been modified in EditContext they all shall have status Good_Edited.~~

~~It is recommended to disallow writes to dependent Variables until after the dominant Variable is applied to the Device.~~

~~2) A change is made in Online EditContext and then to the online Device before the change in the EditContext is applied.~~

~~• The Value in the EditContext will be cleared.~~

~~• The Client monitoring the dominant value and any dependent Variables in the EditContext will get the StatusCode "Good" with no sub-status.~~

~~3) Offline is the same as Online except as follows:~~

~~• The dependent Variables will remain writable before the dominant Variable is applied to the Device.~~

~~b) Change to a dependent Variable:~~

~~1) A change is made in Online EditContext:~~

~~• The Client monitoring Variables in the EditContext will get~~

~~– Good_Edited for the changed dependent Variable~~

~~– Good_DependentValueChanged for the dominant Variable.~~

~~• If dependent and dominant Variables have been modified in EditContext they all shall have status Good_Edited.~~

~~– It is recommended to disallow writes to the dominant Variable until after the dependent Variable is applied to the Device.~~

~~2) A dependent Variable is changed in the Online Device before the change in the EditContext is applied~~

- ~~The Value in the EditContext will be cleared.~~
- ~~The Client monitoring Variables in the EditContext will get~~
  - ~~Good for the cleared dependent Variable and for the dominant Variable.~~

~~3) The dominant Variable is changed in the Online Device before the change of the dependent Variable in the EditContext is applied~~

- ~~The Value of the dependent Variable in the EditContext will be cleared.~~
- ~~The Client monitoring the dominant value and any dependent Variables in the EditContext will get the StatusCode "Good" with no sub-status.~~

~~4) Offline is the same as Online except as follows:~~

- ~~Dominant and dependent Variables are individually writable~~

In some cases, the value of a Variable depends on the value of another Variable. How these dependencies are evaluated is specified in IEC 61804-4.

When such Variables are edited, the FDI Server shall follow the state diagrams specified in Figure 4 for "Online" and Figure 5 for "Offline". These diagrams specify the states and transitions during the editing process of this kind of Variables. Status is the StatusCode that FDI Clients will receive with the Value when monitoring or reading these Variables with the ContextNodeId. For dependent Variables, any Good or Uncertain StatusCode transfers to an Uncertain_DominantValueChanged and a Bad StatusCode to Bad_DominantValueChanged. For dominant Variables, a Good StatusCode transfers into Good_DependentValueChanged, an Uncertain StatusCode into Uncertain_DependentValueChanged and a Bad StatusCode to Bad_DependentValueChanged.

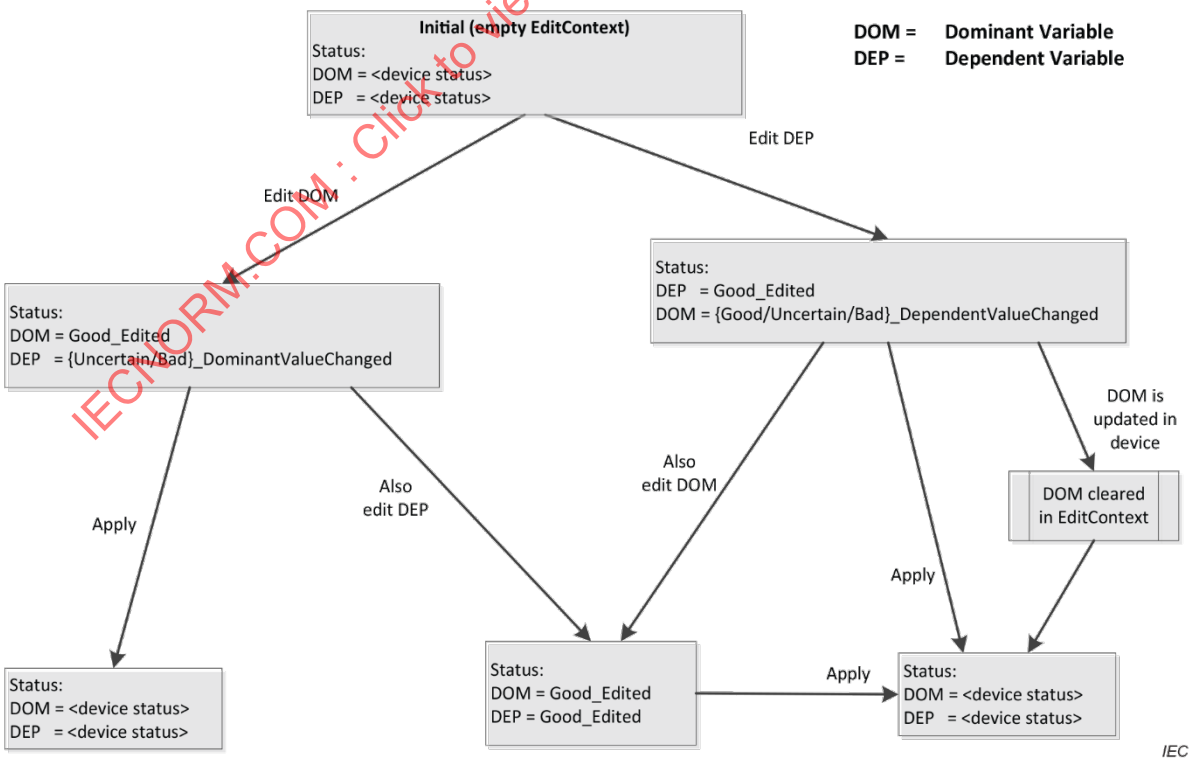

Figure 4 – Online EditContext state diagram for dominant and dependent Variables

If both the dominant and the dependent Variable are to be changed, it is highly recommended for the online case to make these changes in subsequent editing sessions. Systems may enforce this.

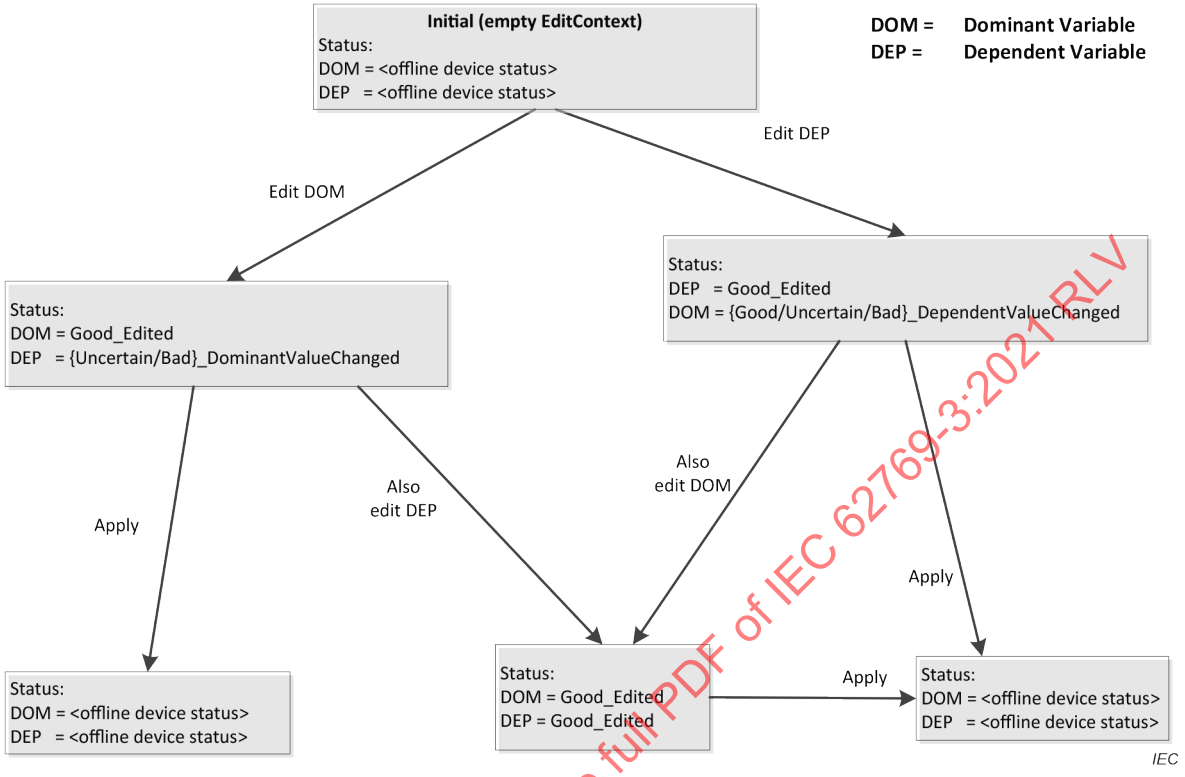

Figure 5 – Offline EditContext state diagram for dominant and dependent Variables

5.6.7 Actions (EDD METHODS)

Before invoking Actions, the Client has to register the ActionSet Node of the Device. The NodeId of this Node has to be specified when calling InvokeAction.

Calling InvokeAction with the ContextNodeId of the ActionSet Node associates it with the proper EditContext instance. The Server will implicitly create an EditContext instance for the invoked Action. This is illustrated in Figure 6.

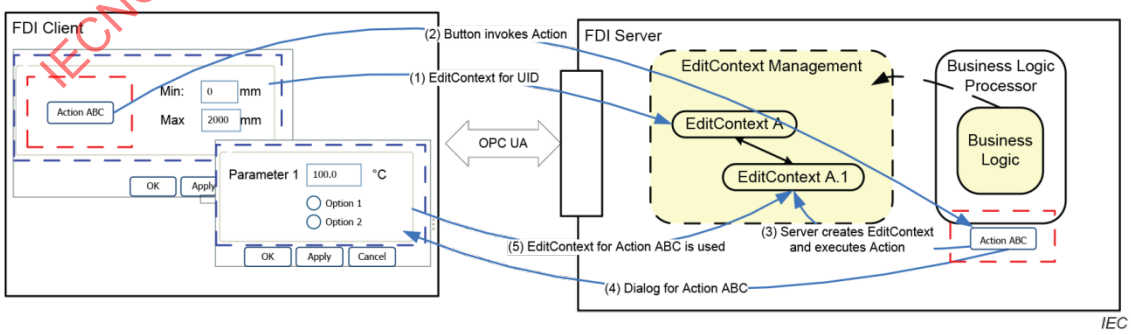

Figure 6 – EditContext for EDD Methods

The EDD METHOD represented by the Action uses builtins to modify Values in the EditContext or the Device, to synchronize changes with the underlying cache or to discard them.

If the Action execution fails, the EditContext for the Action is discarded.

5.6.8 UIDs

The UID Interpreter in the Client calls GetEditContext before it calls up a top-level UID. Additional EditContexts for dialogs may be instantiated by the Server and passed to the Client inside each UID document. See IEC 62769-2 for the UID Schema and the handling of an EditContext in the UID Interpreter.

5.6.9 Synchronization

A Lock has to be created before the first Value is written to an EditContext.

Locking is also required when writing directly to the Device.

5.7 Reading

5.7.1 General

The Read service specified in IEC 62541-4 can be used to read a single value or multiple values from a single device or multiple devices. If a Read service request specifies multiple values are to be read, the values are read in the order they appear in the service request.

All values that are returned to the FDI Client as result of a Read service request shall be unscaled.

A failure encountered while reading a single value shall not abort the read process; all values shall be read. Each value returned contains a status indicating success or failure. Standard OPC UA service status information is returned by the FDI Server as a result of the service calls as specified in 6.2.

An FDI Package can define read actions that are executed by the FDI Server during read service requests. These actions shall not require user interaction; they are strictly intended to be used for Business Logic processing. Any read action that eventually requires user interaction will not perform the builtin but will return an error if possible. The following read actions may be defined in an FDI Package:

- Pre-read Actions
- Post-read Actions

The FDI Server invokes pre-read and post-read actions during the processing of read service requests of online values only; they are not invoked when reading offline values.

In addition to read actions, an FDI Package can define refresh actions that are executed by the FDI Server during read service requests. These actions shall not require user interaction; they are strictly intended to be used for Business Logic processing. Any refresh action that eventually requires user interaction will not perform the builtin but will return an error if possible.

The FDI Server invokes refresh actions during the processing of read service requests of both offline and online values.

The refresh actions mentioned in 5.7.1 are not to be mixed up with refresh relations.

NOTE Refresh actions are defined by means of the EDDL REFRESH_ACTIONS construct inside an EDDL VARIABLE construct. On the other hand, refresh relations are defined by means of the EDDL REFRESH construct. The handling of refresh relations is included in the generic event "Process Conditionals/Relations" that appear in the

sequence diagrams for read, write and subscription services. The explanations that follow the diagrams include refresh relations in the general term "EDDL relations". See IEC 61804-3 for more details on both refresh actions and refresh relations.

5.7.2 Reading offline variables

The sequence diagram in Figure 7 shows the behavior of the FDI Server when an offline value is read.

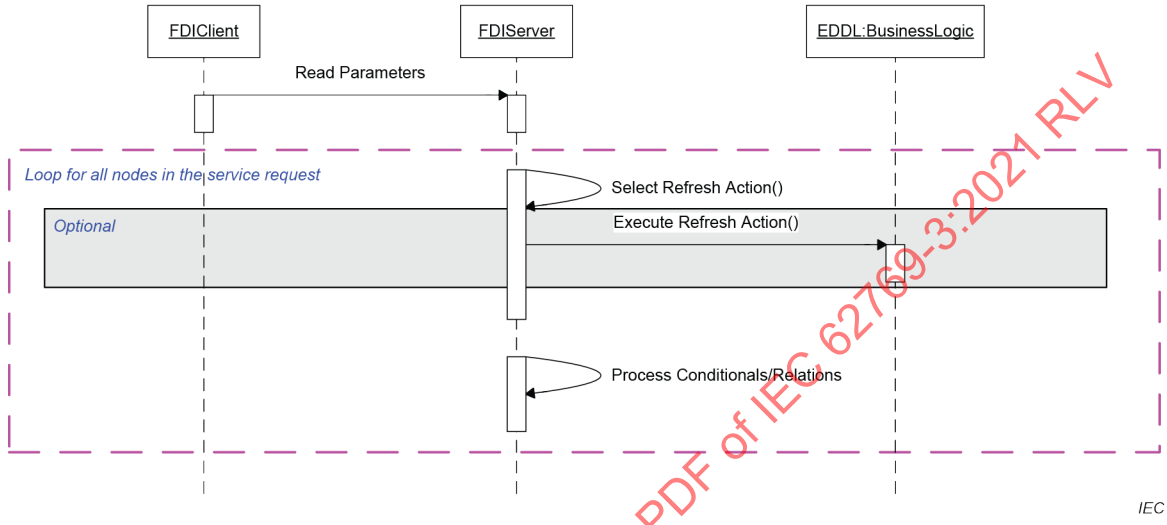

Figure 7 – Offline variable read

If a variable has refresh actions associated with it, the FDI Server always executes those actions regardless of MaxAge.

If the refresh actions fail, the status returned for that variable shall indicate the read failed.

The FDI Server evaluates conditionals and relations after the refresh actions are executed. This provides an opportunity for the re-evaluation of conditional expressions in EDDL Business Logic and the processing of EDDL relations.

5.7.3 Reading online variables

The FDI Server can cache the online values read from a device. The FDI Server maintains a timestamp for each online value that indicates when the value was read from the device. The FDI Server uses the MaxAge argument of a Read service request to determine whether the cached value can be returned. If the difference between the timestamp and the current time exceeds the MaxAge argument, the FDI Server shall read the value from the device. Otherwise, the cached value can be returned.

Read actions are only executed when the variable is read from the device. ~~Read actions are not executed when cached values are returned.~~

The sequence diagram in Figure 8 shows the behavior of the FDI Server when an online value is read.

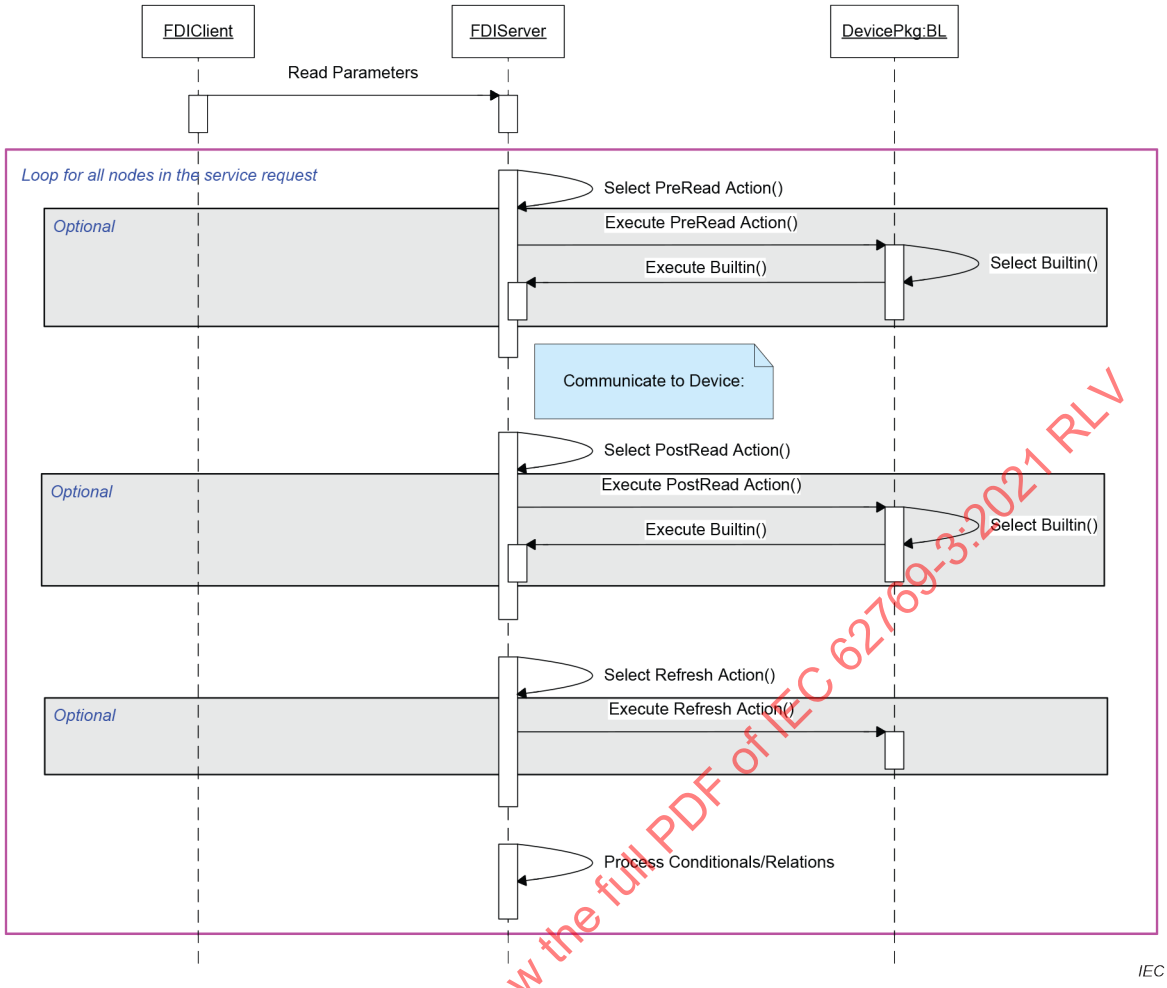

Figure 8 – Online variable read

If a variable has pre-read actions associated with it, these actions are executed prior to reading the variable from the device. If a variable has post-read actions associated with it, these actions are executed after reading the variable from the device. If the pre-read or post-read actions fail, the status returned for that variable shall indicate the read failed.

If a variable has refresh actions associated with it, these actions are handled as in the offline variable read case (see 5.7.2).

The FDI Server evaluates conditionals and relations after post-read actions are executed. This provides an opportunity for the re-evaluation of conditional expressions in EDDL Business Logic and the processing of EDDL relations.

5.8 Writing

5.8.1 General

The Write service specified in IEC 62541-4 can be used to write a single value or multiple values to a single device or multiple devices. If a Write service request specifies multiple values are to be written, the values are written in the order they appear in the service request.

A failure encountered while writing a single value shall not abort the write process; all values shall be written. A status is returned indicating success or failure of each value included in the service request. Standard OPC UA service status information is returned by the FDI Server as a result of the service calls as specified in 6.2.

Unlike the read operation, write failures when multiple variables are specified may leave the device in an indeterminate state with some variables modified and others left unmodified. It is up to the FDI Client to handle partial failures.

FDI Clients need to lock the device for exclusive access prior to writing. The lock request may be issued immediately before the write service request or it may be issued independently across multiple write service requests (see 5.5).

The FDI Server performs data validation during write service requests of online and offline values.

An FDI Package can define write actions that are executed by the FDI Server during write service requests. These actions shall not require user interaction; they are strictly intended to be used for Business Logic processing. Any write action that eventually requires user interaction will not perform the builtin but will return an error if possible. The following write actions may be defined in an FDI Package:

- Pre-write Actions
- Post-write Actions

The FDI Server invokes those actions during the processing of write service requests of online values only; they are not invoked when writing offline values.

5.8.2 Write offline variables

The sequence diagram in Figure 9 shows the behavior of the FDI Server when an offline value is written.

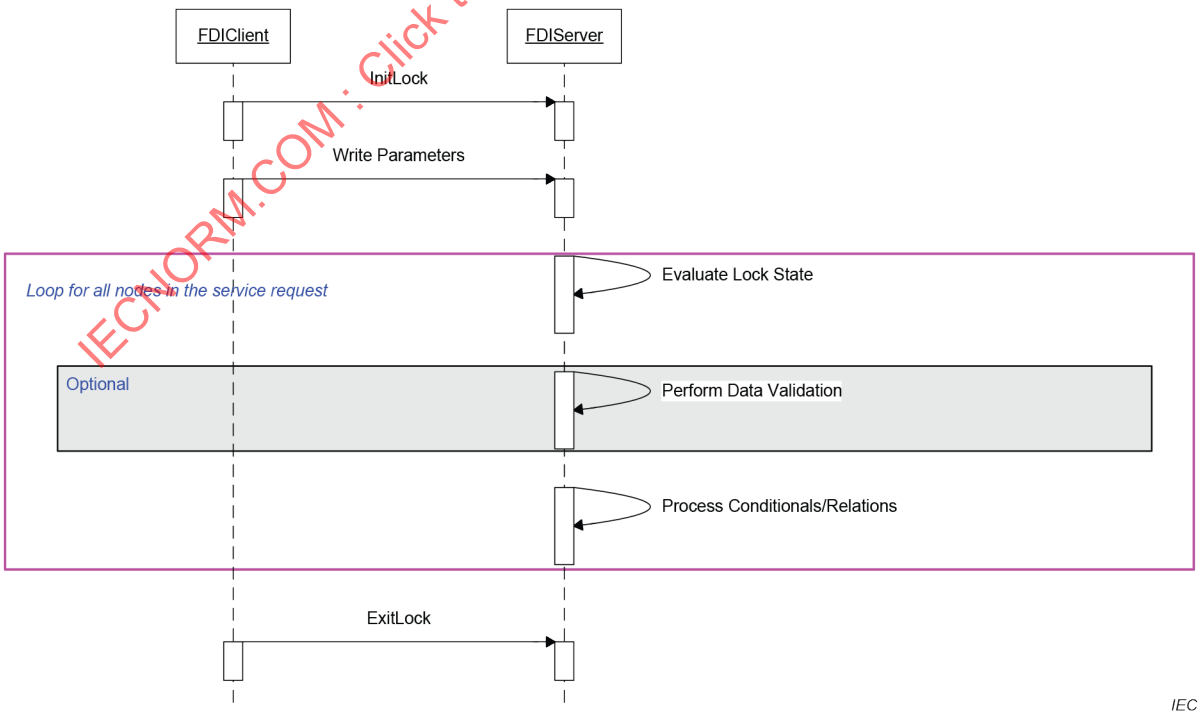

Figure 9 – Offline variable write immediate

As a starting point for writing a variable, the FDI Server verifies if the device is locked by the FDI Client. If it is not locked, the status returned for that variable shall indicate the write failed.

If the device is locked by the FDI Client, the FDI Server performs data validation. The validation consists basically of range and type check based on EDDL information. If the type validation fails, the status returned for that variable shall indicate the write failed. If the range validation fails, the status returned for that variable shall indicate the write succeeded, but the status information of the variable value in the Information Model shall indicate that it is bad, out-of-range.

If the validation process succeeds, the FDI Server writes to the offline value of the variable in the Information Model.

After writing the variable value, the FDI Server evaluates conditionals and relations. This provides an opportunity for the re-evaluation of conditional expressions in EDDL Business Logic and the processing of EDDL relations.

5.8.3 Writing online variables

The sequence diagram in Figure 10 shows the behavior of the FDI Server when an online value is written.

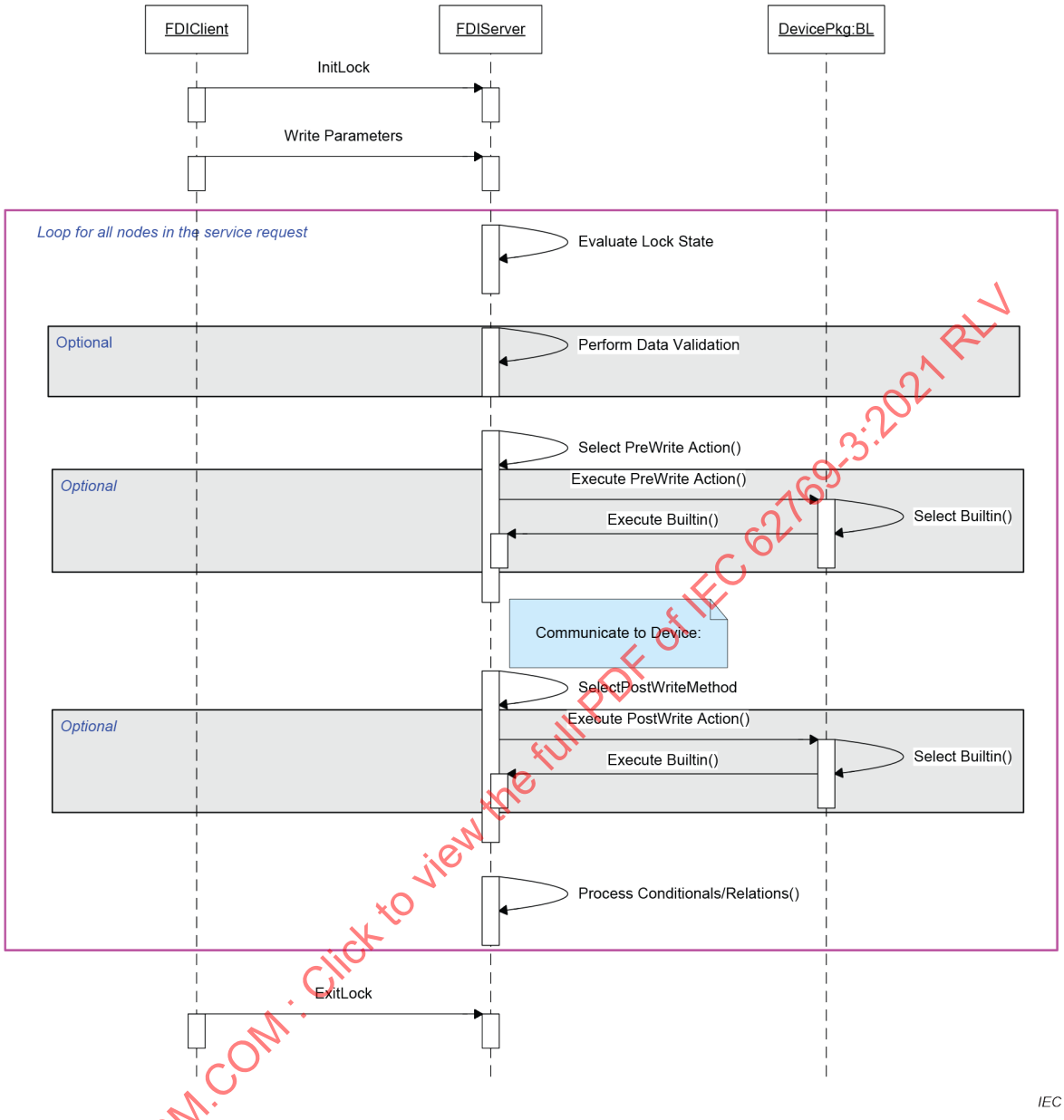

Figure 10 – Online variable write immediate

When writing an online variable, the FDI Server verifies if the device is locked by the FDI Client and performs data validation as described in 5.8.2.

If the validation process succeeds, the FDI Server writes the variable to the physical device.

If a variable has pre-write actions associated with it, these actions are executed prior to writing the variable to the device. If the pre-write actions fail, the status returned for the variable shall indicate the write failed and write operation terminates without writing to the device.

If a variable has post-write actions associated with it, these actions are executed after writing the variable to the device. If the post-write actions fail, the status returned for the variable shall not indicate the write failed, since the value has already been written to the device. The status returned shall be Good_PostActionFailed.

The FDI Server evaluates conditionals and relations after post-write actions are executed. This provides an opportunity for the evaluation of conditional expressions in EDDL Business Logic and the processing of EDDL relations.

5.8.4 Writing to an EditContext

The EditContext is specified in 5.6.

The sequence diagram in Figure 11 shows the general behavior of an EditContext when Values are edited and applied.

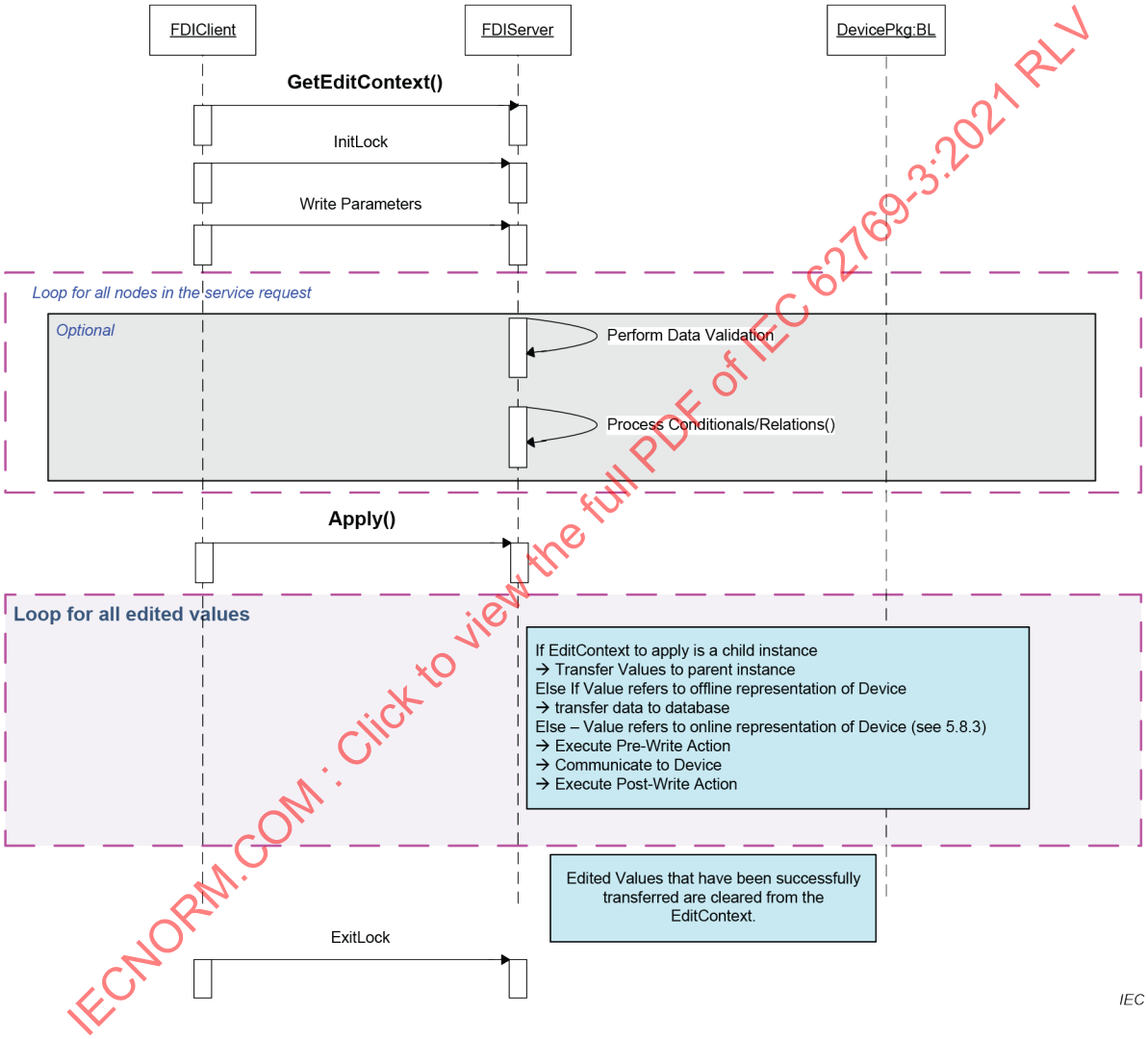

Figure 11 – Write with EditContext

When writing Variables to an EditContext, the FDI Server performs data validation as in the normal writes to online or offline data. It also processes Conditionals/Relations. Changes to other Variables resulting from this process are also written to the EditContext.

When calling Apply, the modified Variables are transferred to the parent. If the parent is the Device and a variable has pre-write actions associated with it, these actions are executed prior to writing the variable to the Device. If the pre-write actions fail, the status returned for Apply shall indicate the error.

If the parent is the Device and a variable has post-write actions associated with it, these actions are executed after writing the variable to the Device. If the post-write actions fail, the status returned for the variable shall not indicate the write failed, since the value has already been written to the device. The status returned shall be Good_PostActionFailed.

5.9 Subscription

5.9.1 General

The Subscription service specified in IEC 62541-4 can be used to initiate the monitoring of a single variable or multiple variables from a single device or multiple devices.

A failure related to a single variable while establishing a subscription to multiple variables shall not abort the subscription process; all variables shall be monitored. Each variable returned contains a status indicating success or failure. Standard OPC UA service status information is returned by the FDI Server as a result of the service calls, as specified in 6.2.

An FDI Package can define read actions that are executed by the FDI Server during the monitoring process of a subscription. These actions shall not require user interaction; they are strictly intended to be used for Business Logic processing. Any read action that eventually requires user interaction will not perform the builtin but will return an error if possible. The following read actions may be defined in an FDI Package:

- Pre-read Actions
- Post-read Actions

The FDI Server invokes these actions during the monitoring of online variables only; they are not invoked when monitoring offline variables.

In addition to read actions, an FDI Package can define refresh actions that are executed by the FDI Server during the monitoring process. The FDI Server invokes these refresh actions during the monitoring of both offline and online variables. These actions shall not require user interaction; they are strictly intended to be used for Business Logic processing. Any refresh action that eventually requires user interaction will not perform the builtin but will return an error if possible.

The sampling interval requested by the FDI Client and established by the FDI Server defines a time interval that is used to periodically check for changes in the variables' value or status. At each time interval, the actions are invoked, and the value and status are compared with the previous value and status. A change in the value or status will result in the FDI Server preparing a notification of the new value and status.

5.9.2 Subscription of offline variables

The sequence diagram in Figure 12 shows the behavior of the FDI Server when an offline value is being monitored.

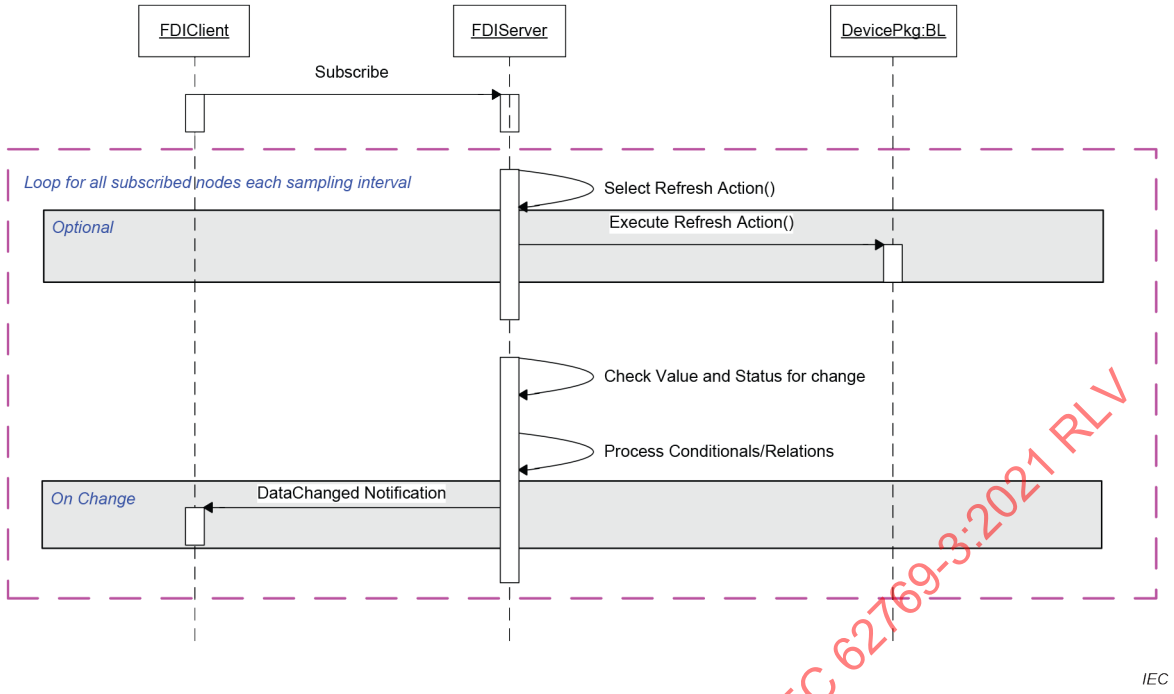

Figure 12 – Offline variable subscription

If a variable has refresh actions associated with it, the FDI Server executes those actions at each time interval.

The FDI Server evaluates conditionals and relations after the refresh actions are executed.

5.9.3 Subscription of online variables

The sequence diagram in Figure 13 shows the behavior of the FDI Server when an online variable is being monitored.

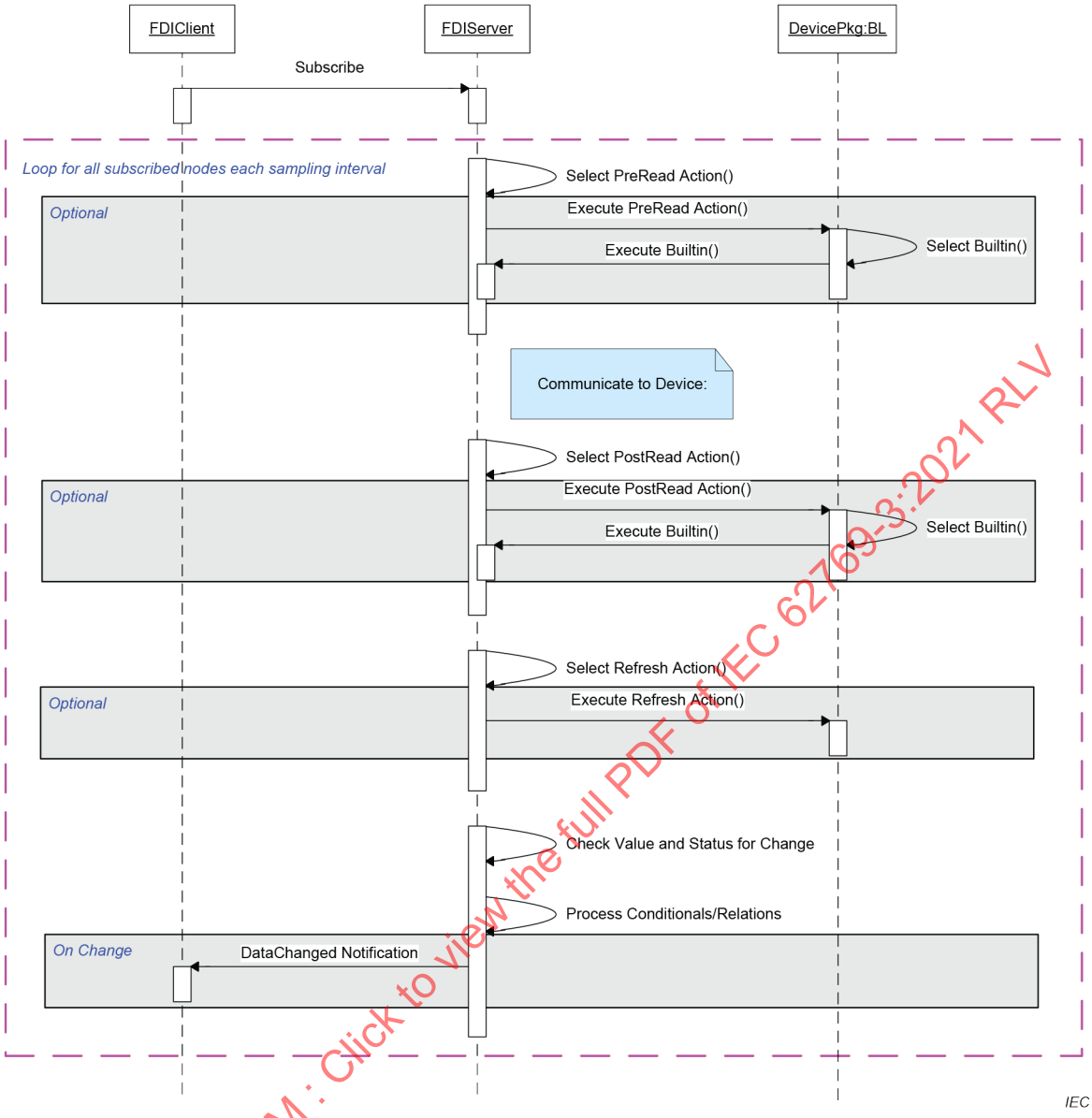

Figure 13 – Online variable subscription

The variable is read from the device at each time interval.

If a variable has pre-read actions associated with it, these actions are executed prior to reading the variable from the device.

If a variable has post-read actions associated with it, these actions are executed after reading the variable from the device.

If a variable has refresh actions associated with it, the FDI Server executes those actions after the post-read actions.

The FDI Server evaluates conditionals and relations after any associated actions are executed.

If the actions fail or the device read fails, the status of the variable shall indicate the failure.

5.10 Device topology

5.10.1 General

The FDI Server maintains Device Instances in the Information Model. The Information Model maintained by the FDI Server reflects the structure of the system; the FDI Server maintains device information in the context of the Device Topology.

The Device Topology includes devices, connecting communication networks, and the elements to communicate via these networks. The Device Topology is defined in IEC 62769-5; Objects, References and the AddressSpace organization required to create the proper Information Model are defined as part of the Information Model specification.

5.10.2 Connection Points

The following non-normative Figure 14 illustrates the topology within the Information Model.

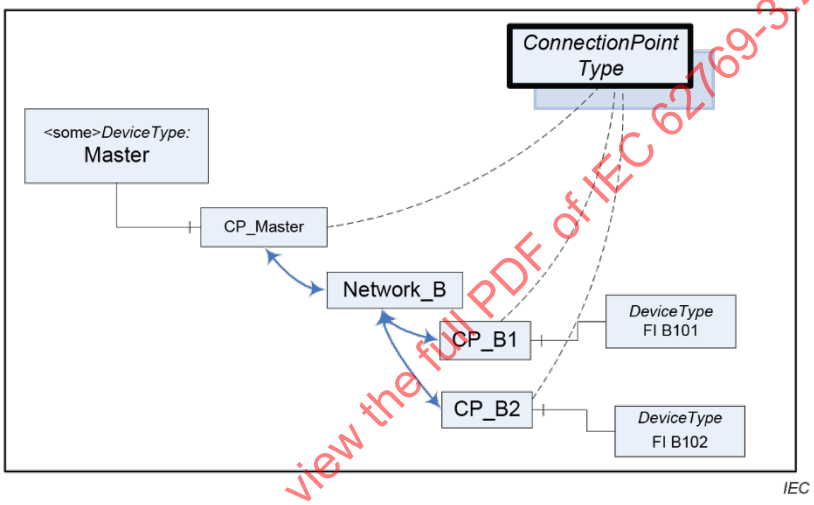

Figure 14 – Topology with Network objects (non-normative)

The FDI Server uses the information in the FDI Package to create both the type definitions for the devices and Communication Devices as well as their respective Connection Point elements. The mapping between the FDI Package definition and the Information Model elements is defined in IEC 62769-5.

Network types for the protocols that are supported by Native Communication devices are provided by the FDI Server. Network types for non-native protocols are provided through FDI Package definitions provided for the communication server.

Device definitions contain one or more Connection Point definitions for a device. Each Connection Point maintains a reference to a protocol element that specifies the protocol for the Connection Point. A device may be capable of providing or using multiple protocols; each protocol provided or supported will have a unique Connection Point. The network objects contain a reference to a protocol definition element that defines the protocol utilized by the network object.

The Device and Connection Point type definitions are used to create Device and Connection Point instances. The network type definitions are used to create instances of networks in the system. The network types are specified in the protocol-specific annexes in IEC 62769-4, and are provided by the FDI Server as an integral component. The FDI Server, or FDI Clients interacting with the FDI Server to create the topology, locates devices, communication devices, connection points, and networks using standard OPC UA entry points for browsing:

- DeviceSet – references Device Instances in the FDI Server;
- CommunicationSet – references instances of communication devices in the FDI Server;
- NetworkSet – references instances of networks in the FDI Server.

The FDI Server, or FDI Clients interacting with the FDI Server to create the topology, creates references between instances of a device, an associated Connection Point, and a network element (see Figure 14). The FDI Server validates that the protocol associated with the Connection Point and the protocol associated with the network are of the same protocol type. If a device defines multiple Connection Points, the FDI Server will use the Connection Point of the device that matches the network protocol.

Each network object shall be associated with at least one Communication Device. The association between the communication device and the network element shall be done before devices can be associated with the network element. Once a Communication Device is associated with a network element, Business Logic in the Communication Device will be used for network management.

The network element definition specifies the number of Connection Points that can be added to the network.

The references established between devices, Connection Points, and networks do not affect the reference established for the standard browse entry points. Devices remain referenced by DeviceSet regardless of whether the device has a reference to a Connection Point or to a network.

5.10.3 Topology management

5.10.3.1 General

FDI Server vendors have two options to provide trusted FDI Clients with the ability to manage the topology:

a) they may provide vendor-specific functionality;

b) they may implement the OPC UA NodeManagement Service Set.

If the FDI Server vendor chooses the second option, i.e. implementing the OPC UA NodeManagement Service Set, the topology management shall be implemented as specified in 5.10.3.

In order to prevent simultaneous access from different agents that are trying to modify the topology, the elements involved in the topology modification are locked. The scope of the lock for Modular Devices and networks is specified in IEC 62769-5.

The FDI Package for the Communication Device includes definitions that are used by the FDI Server to manage and validate the topology, including the optional action ValidateNetwork (see IEC 62769-7). Those definitions are used by the FDI Server during topology management.

5.10.3.2 Add Device to Network

The sequence diagram in Figure 15 shows the behavior of the FDI Server when a device is added to a network.

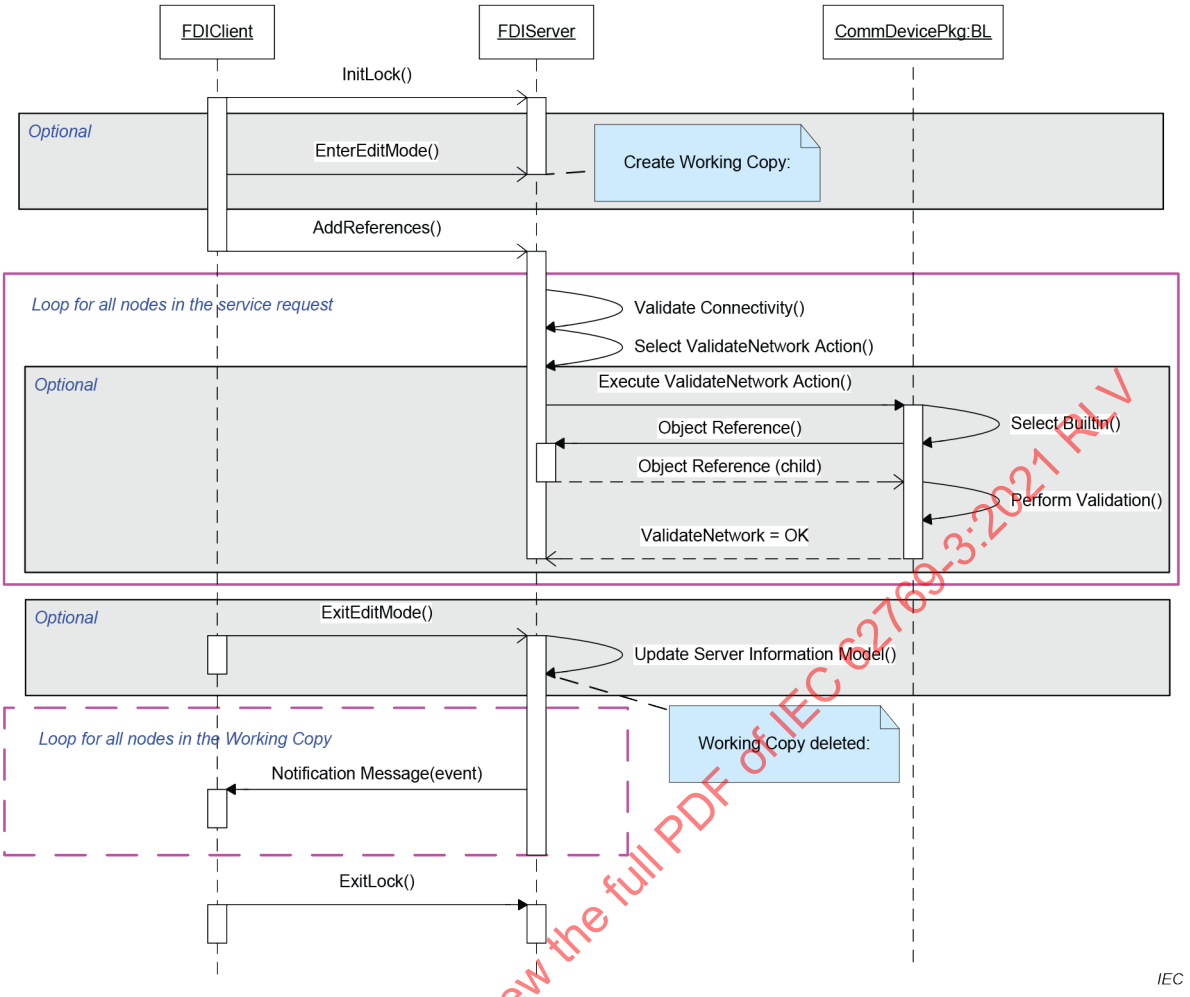

Figure 15 – Add Device to topology

The AddReferences service specified in IEC 62541-4 is used to add devices to the topology. The AddReferences service can be used to establish a single or multiple references. If an AddReferences service request specifies multiple references are to be added, the references are added in the order they appear in the service request.

A failure encountered while adding a single reference shall not abort the entire process; all references shall be processed. A status is returned indicating success or failure for each reference included in the service request. Standard OPC UA service status information is returned by the FDI Server as a result of the service calls, as specified in 6.2.

When adding a reference, the association of a device's Connection Point with a network shall be validated by the FDI Server; this applies to both FDI Server vendor-specific topology management as well as the OPC UA AddNode method.

The FDI Server performs an initial validation of the connectivity, which includes, for instance, verifying that the protocol specified by the Connection Point and the network are the same and the number of connections supported by the network have not been exceeded. That validation is based on information provided with the FDI Communication Package of the involved elements. If the initial validation succeeds, the ValidateNetwork Action provided by the Communication Device is invoked by the FDI Server. See IEC 62769-7 for more details.

In a first pass, all requested references are added regardless of the validation process succeeding or failing. After adding all references, a second validation pass is done. If the

validation process fails for any reference, that reference shall not be added, and the status returned for that reference shall indicate the reference failed to be added.

Because of this two-step validation process performed by the FDI Server, the result of adding multiple references at a time can be different from the result of adding one reference at a time.

5.10.3.3 Remove Device from Network

The sequence diagram in Figure 16 shows the behavior of the FDI Server when a device is removed from a network.

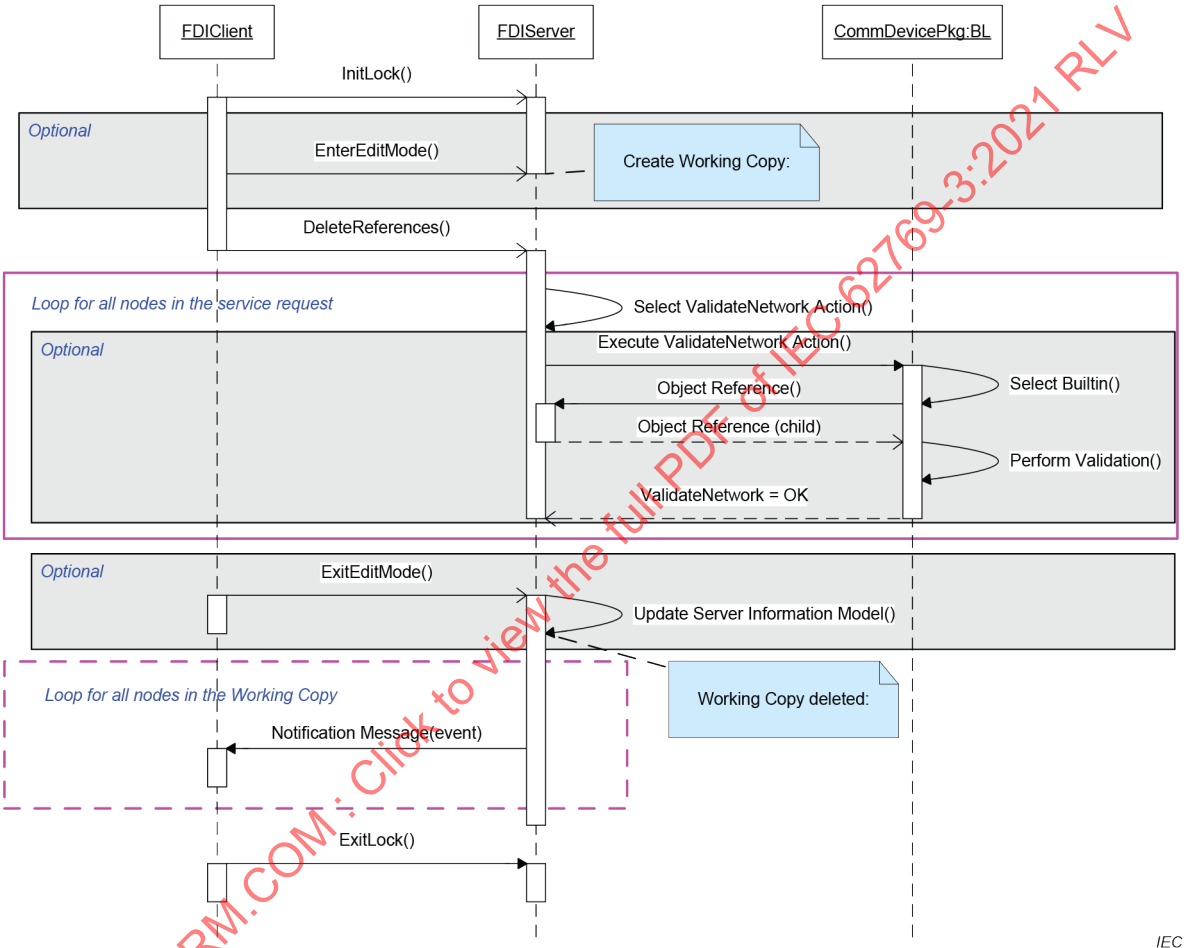

Figure 16 – Remove device from topology

The DeleteReferences service specified in IEC 62541-4 is used to remove devices from the topology. The DeleteReferences service can be used to remove a single reference or multiple references. If a DeleteReferences service request specifies multiple references are to be removed, the references are removed in the order they appear in the service request.

A failure encountered while removing a single reference shall not abort the entire process; all references shall be processed. A status is returned indicating success or failure for each reference included in the service request. Standard OPC UA service status information is returned by the FDI Server as a result of the service calls, as specified in 6.2.

The removal of a device from a network requires the FDI Server to validate the network. The FDI Server invokes the ValidateNetwork Action provided by the Communication Device after removal of the device from the network.

In a first pass, all requested references are removed regardless of the validation process succeeding or failing. After removing all references, a second validation pass is done. If the validation process fails for any reference, that reference shall not be removed, and the status returned for that reference shall indicate the reference failed to be removed.

Because of this two-step validation process performed by the FDI Server, the result of adding multiple references at a time can be different from the result of adding one reference at a time.

5.10.4 Topology scanning

The sequence diagram in Figure 17 shows the behavior of the FDI Server when the topology is scanned.

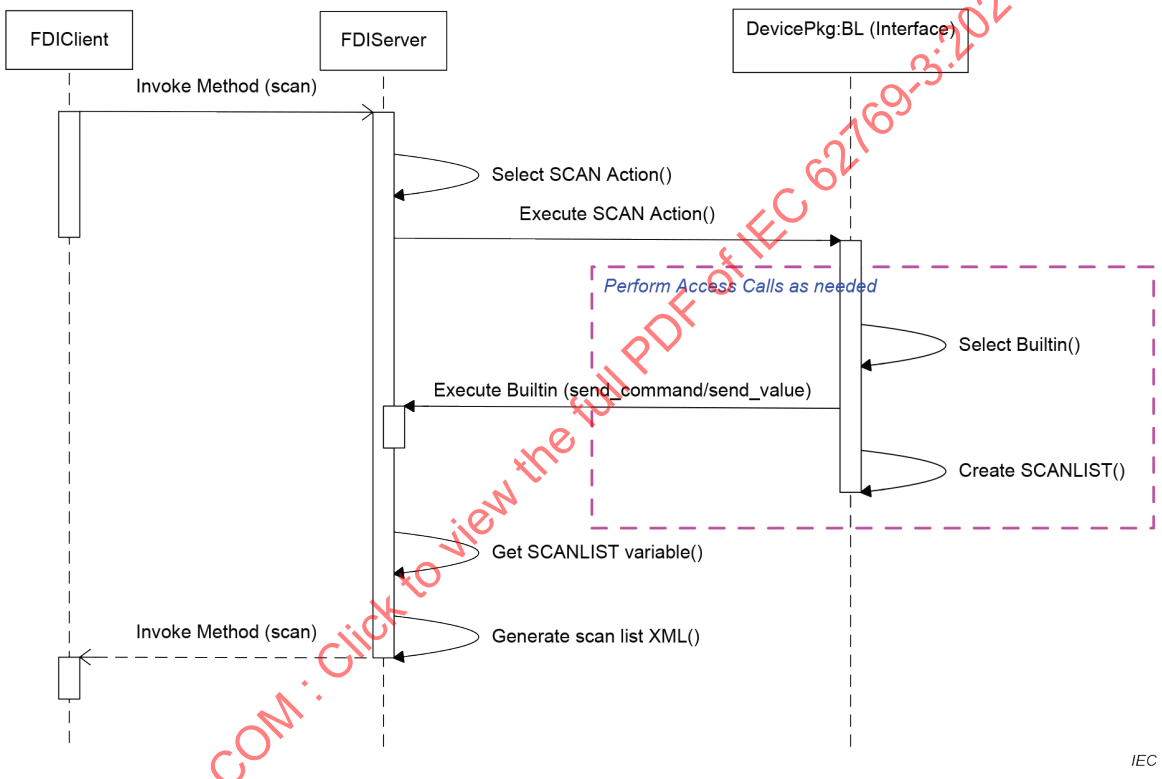

Figure 17 – Scan topology

Scanning of a network for connected devices is provided via the Scan Service associated with Communication Devices. This service is described in IEC 62769-5.

For nodes representing Native Communication devices, the FDI Server provides the service implementation.

For Communication Devices that are not native to the FDI Server, the FDI Server invokes the SCAN Action provided by the Communication Device (see IEC 62769-4). The SCAN action invokes builtins provided by the FDI Server to send commands to the Communication Device. The SCAN action processes the responses to the commands to create a scan list.

The scan list created by the SCAN Action is stored in a DDLIST variable referenced through the SCAN_LIST variable. The DDLIST contains the definition for the devices detected by the communication device.

The Information Model specified in IEC 62769-5 provides protocol independent definitions for devices. The protocol independent device definitions contain references to nodes containing protocol-specific identification for a device.

The DDLIST variable resulting from the scan is composed of variable definitions. The DDLIST will contain variables whose name matches the properties and attributes defined in IEC 62769-5 for the protocol independent device definition. These protocol-independent definitions allow the FDI Server to formulate protocol-independent information to an FDI Client about the devices in the network.

The SCAN Action may not fully identify a device; variables providing protocol-independent information may be missing from the DDLIST. The FDI Server through vendor-specific functions performs additional parameter read and write actions to the device to complete the identification. This functionality is both FDI Server vendor-specific as well as protocol-specific and is not standardized.

The DDLIST also contains network addressing information for the devices identified in the network. The network addressing will be protocol-specific but match the protocol-specific Connection Point properties and attributes specified in IEC 62769-5.

The DDLIST may also contain additional variables that are protocol-specific. The protocol-specific variables are standardized by the foundations defining the network protocol, see protocol-specific annexes in IEC 62769-4.

The FDI Server is responsible for the creation of the XML data set returned by the Scan Service using the information provided by the DDLIST variable.

5.10.5 Use of SCAN function

The SCAN function can be used by the FDI Server as part of topology management. The FDI Server vendor-specific functions for topology management may perform network SCANS to define the Device Instances to create and to initialize the Information Model topology.

The SCAN function is provided by communication devices; a device definition does not have to exist in the Information Model for the SCAN function to succeed. The information provided by the SCAN function may be used by FDI Server vendor-specific functionality to create the Device Topology. FDI Server vendor-specific functionality is responsible for matching the devices identified by the SCAN function to device types in the Information Model.

The FDI Server, through vendor-specific implementation, uses the SCAN function as part of commissioning a network. The FDI Server vendor-specific implementation allows the FDI Server to match and validate the offline created topology against the physical network (see 5.10.6).

The use of the SCAN function for topology creation and topology validation is not standardized and is an FDI Server vendor-specific functionality.

5.10.6 Validation of defined topology

The FDI Server shall validate that the defined Device Instance in the Information Model matches the physical device. The FDI Server vendor-specific functionality is responsible for validating the Information Model against the physical devices connected in the system.

The FDI Server can rely on standard functions such as SCAN to identify the devices physically connected and determine whether there is a match with the offline defined topology. The FDI Server may also use protocol-specific commands to identify devices.

The FDI Server shall validate that the physical device is of the same type as the Device Instance in the Information Model. The FDI Server vendor-specific implementations can allow

connectivity to devices of different revisions or require an exact match. FDI Server vendor-specific functionality provides the device matching.

5.11 User Interface Elements

5.11.1 User Interface Descriptions

User Interface Descriptions (UIDs) are descriptive user interfaces that are rendered by an FDI Client. They appear in the Information Model as UIDescriptionType nodes (see IEC 62769-5).

FDI Clients retrieve UIDs by reading the Value attribute of a UIDescriptionType node in the Information Model. The Value attribute of a UIDescriptionType node contains the UID in the form of an XML string (see IEC 62769-2).

Any values that the FDI Server provides to the FDI Client through the UID shall be unscaled, including, for instance, current variable values, ranges and initial values.

FDI Servers shall evaluate any conditional behavior present in a UID before providing it to the FDI Client. The FDI Client shall simply render the UID provided by the FDI Server.

This example assumes the UID is based on EDDL. If the PATH attribute of an IMAGE is conditional (i.e., dependent on the value of a parameter in the device), the FDI Server shall evaluate the conditional to determine which image is to be used and then provide the appropriate Browse Name to the FDI Client via the XML string. The FDI Client will simply render the appropriate image. All other EDDL conditionals shall be evaluated by the FDI Server in a similar fashion.

An FDI Package can define actions that are associated with UIDs. The following actions may be defined in an FDI Package:

- Pre-edit Actions;
- Post-edit Actions;
- Init Actions;
- Refresh Actions;
- Exit Actions.

Those actions are executed by the FDI Server, but their execution is driven by the FDI Client.

In order to allow the FDI Client to drive the execution of actions, the FDI Server creates Actions Proxies (see 5.12.3) and makes the Action Proxies names available to the FDI Client by means of the ListOfActions element type defined in the XML schema (IEC 62769-2). The FDI Server thus maintains the Actions Proxies names in the XML string of the UID.

As the FDI Client retrieves UIDs, it retrieves the Actions Proxies names associated to them as well.

Even though a single EDD Actions definition may specify more than one EDD Method, the FDI Server does not provide individual references to each EDD Method that is specified, but it provides a single Actions Proxy name to refer to all EDD Methods specified in the EDD Actions clause. As a consequence, the list of actions specified in the XML schema will always have a single entry.

As the FDI Client processes a UID, it can start the execution of actions by calling the InvokeAction service and passing the corresponding Actions Proxy name as argument (see 5.12.3).

5.11.2 User Interface Plug-ins

User Interface Plug-ins (UIPs) are programmatic user interfaces that are executed by an FDI Client. They appear in the Information Model as UIPluginType nodes (see IEC 62769-5).

FDI Clients retrieve UIPs by reading the Value attribute of a UIPluginType node in the Information Model. The Value attribute of a UIPluginType node is a byte array containing a binary executable component (see IEC 62769-5).

FDI Packages can provide multiple variants of the same UIP (see IEC 62769-4). FDI Clients browse through the available UIP Variants and select the variant that is most appropriate.

Unlike UIDs, UIPs are not processed by an FDI Server in any way; they are imported from the FDI Package and simply provided to the FDI Client upon request.

FDI Device Packages can reference a UIP in a separate FDI UIP Package (see IEC 62769-4). FDI Servers shall resolve these references. Any references that cannot be resolved shall result in a Bad_NodeIdUnknown status code when the UIP is read by an FDI Client.

5.12 Actions

5.12.1 FDI Server – FDI Client interaction

FDI Clients invoke Actions by calling the InvokeAction method (see IEC 62769-5).

When an Action is invoked the FDI Server creates a state machine that is maintained while the Action is executing. The state may change in response to the builtin functions that are invoked by the Action, as well as in response to interactions with the FDI Client.

The FDI Server then creates a transient, non-browsable Variable in the Information Model for the exchange of information between the FDI Server and the FDI Client, henceforth referred to as the exchange variable. The NodeId of the exchange variable is returned to the FDI Client as an output argument of the InvokeAction method (see IEC 62769-5).

Once the state machine has been created, the exchange variables have been created, and the Action has started to execute, the InvokeAction method terminates, i.e. it does not remain active during the execution of the Action.

The FDI Server sends user interface requests to the FDI Client via the exchange variable, and the FDI Client sends user interface responses to the FDI Server via the exchange variable. The value of the exchange variable is an XML string (see IEC 62769-2). It contains the current state of the Action, as well as a user interface request or response.

The subscription service specified in IEC 62541-4 is used to allow the FDI Server to send user interface requests to the FDI Client via the exchange variable. The FDI Client subscribes to the exchange variable to receive user interface requests from the FDI Server. If a request is transitional the FDI Client may miss the request. The request will be held in the exchange variable until the FDI Client creates a subscription. Once the subscription is established, the FDI Server will respond with the current state and the pending request.

NOTE 1 TimeDelay with a short duration is an example of a transitional request.

The FDI Server can implement a server-defined time-out for user interface requests. Failure of the FDI Client to respond to a user interface request before the time-out expires can cause the FDI Server to abort the Action.

NOTE 2 The time-out is expected to be on the order of 20 min to 30 min similar to a session time-out of a web page.

The FDI Server retains the current state and the last request in the exchange variable even after the Action completes. The exchange variable retains its value until the FDI Client terminates the subscription.

An Action can be aborted by the FDI Client or by the Action itself.

The sequence diagram shown in Figure 18 shows the client/server interaction of an Action.

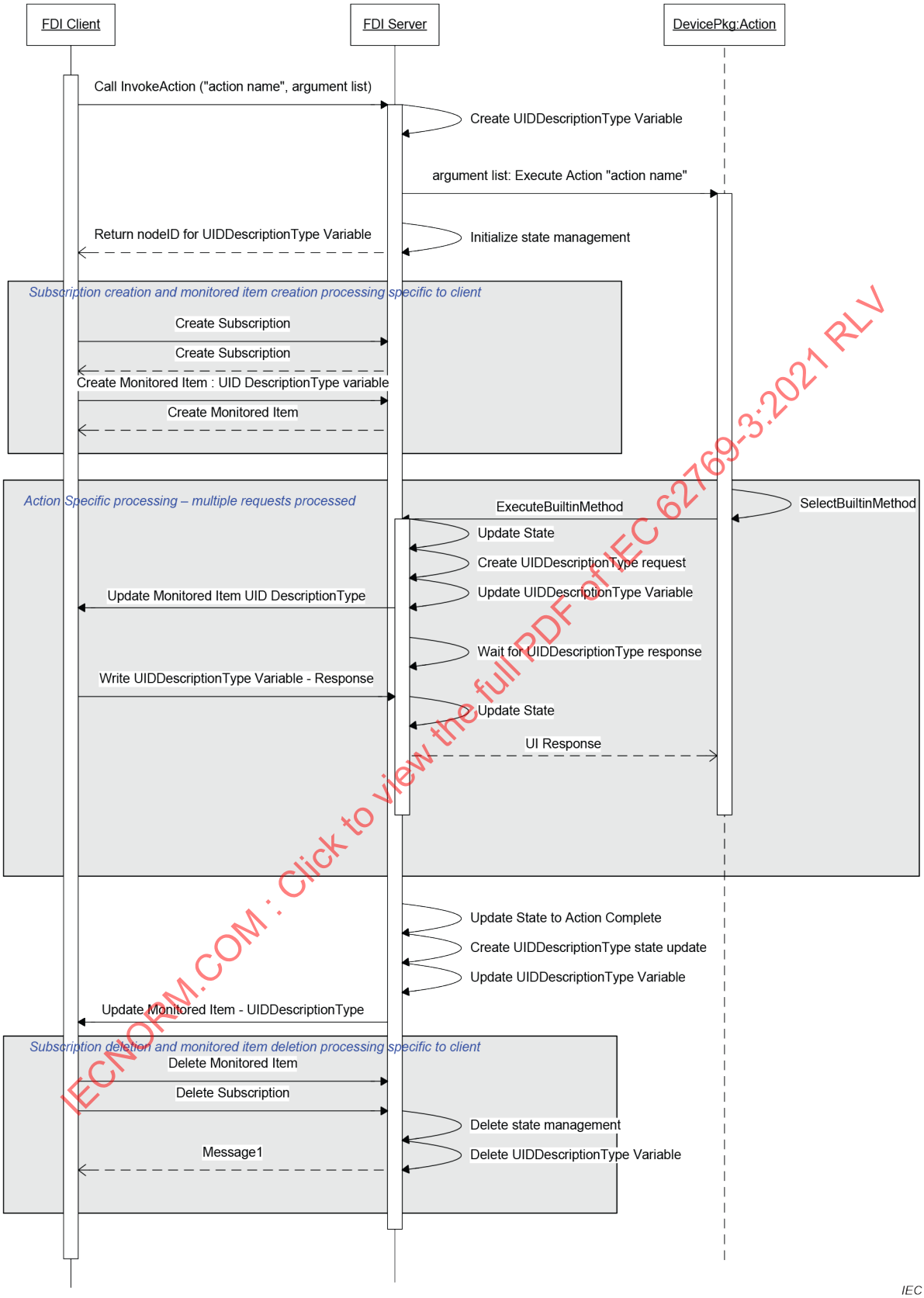

Figure 18 – Action execution

5.12.2 Action state machine

5.12.2.1 States

The Action state machine is shown in Figure 19.

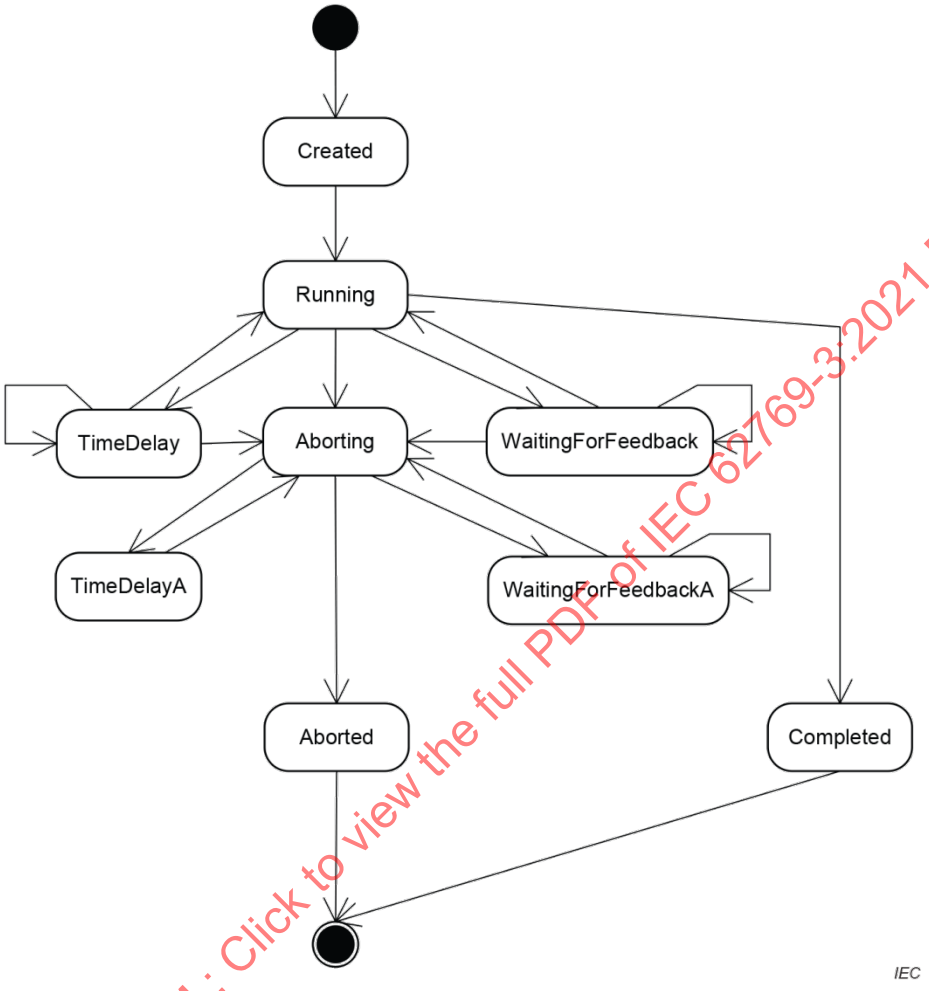

Figure 19 – Action state machine

The states of the Action state machine are specified in Table 1.

Table 1 – Action states

State	Description
Created	The initial state when the state machine instance is created by the FDI Server.
Running	The normal execution state.
TimeDelay	The state where the normal execution is suspended for a certain amount of time.
WaitingForFeedback	The state where the normal execution is suspended because a user interaction is needed.
Aborting	The state where the normal execution has been aborted and abort processing is carried out.
TimeDelayA	The state where the abort processing is suspended for a certain amount of time.
WaitingForFeedbackA	The state where the abort processing is suspended because a user interaction is needed.

State	Description
Completed	The state where the normal execution is completed.
Aborted	The state where the abort processing is completed.

5.12.2.2 State transitions

The state transitions of the Action state machine are defined in Table 2.

Table 2 – Action state transitions

Source state	Event	Destination state
Start State	FDI Server created a state machine for the Action.	Created
Created	Execution of the Action has started.	Running
Running	Execution of the Action has completed.	Completed
Running	Builtin function has been encountered that requires a time delay.	TimeDelay
Running	Builtin function has been encountered that requires user feedback.	WaitingForFeedback
Running	Abnormal termination of the Action has been initiated by either the FDI Server or the FDI Client.	Aborting
TimeDelay	FDI Server has decided to send time delay remaining to FDI Client.	TimeDelay
TimeDelay	FDI Server has calculated a new time delay to be sent to FDI Client.	TimeDelay
TimeDelay	Abnormal termination of the Action has been initiated by the FDI Client.	Aborting
TimeDelay	Delay time has expired.	Running
WaitingForFeedback	FDI Server has decided to send an updated feedback request to the FDI Client.	WaitingForFeedback
WaitingForFeedback	FDI Server has received feedback from FDI Client.	Running
WaitingForFeedback	Abnormal termination of the Action has been initiated by either the FDI Server or the FDI Client.	Aborting
WaitingForFeedback	FDI Server timeout period has expired with no response from FDI Client.	Aborting
Aborting	Builtin function has been encountered that requires a time delay.	TimeDelayA
Aborting	Builtin function has been encountered that requires user feedback.	WaitingForFeedbackA
Aborting	Execution of the Action has completed.	Aborted
TimeDelayA	Delay time has expired.	Aborting
WaitingForFeedbackA	FDI Server has decided to send an updated feedback request to the FDI Client.	WaitingForFeedbackA
WaitingForFeedbackA	FDI Server has received feedback from FDI Client.	Aborting
Aborted	FDI Server has destroyed the state machine for the Action.	Finish State
Completed	FDI Server has destroyed the state machine for the Action.	Finish State

5.12.3 Actions Proxies

EDD Actions specify EDD Methods that shall be executed at specific moments during the processing of variable values or during user interaction. In many cases, the FDI Server implicitly executes the EDD Actions, but in some specific cases, as specified in 5.12.4, the execution of EDD Actions is driven by the FDI Client.

In order to allow the FDI Client to drive the execution of EDD Actions, the FDI Server creates Actions Proxies. An Actions Proxy is an internal entity created by the FDI Server to encapsulate the EDD Methods specified in the EDD Action definition. An Actions Proxy thus corresponds to a single "*_ACTIONS" clause in EDDL and therefore to the entire set of EDD Methods specified in it.

The FDI Server assigns a name to the Actions Proxy. The Actions Proxy name is an unambiguous identifier, i.e. it uniquely identifies the Actions Proxy in the scope of a single device instance.

The FDI Server makes the Actions Proxy name available to the FDI Client via the XML descriptions associated to UID nodes (see 5.11.1). The FDI Client can thus drive the execution of an Actions Proxy when necessary by calling the InvokeAction method and passing the Actions Proxy name as argument.

NOTE The second argument of the InvokeAction method ("MethodArguments") is empty, since EDD Actions have no arguments.

When processing an InvokeAction call with an Actions Proxy name as argument, the FDI Server executes the entire set of EDD Methods associated to that Actions Proxy. As specified in ~~IEC 61804-3~~ a, the FDI Server executes those EDD Methods in the order they appear in the EDD Action definition, and if an EDD Method exits for an unplanned reason, the following EDD Methods are not executed.

Taking as reference the state transitions defined in Table 2, it means that:

- the state machine transitions from the state "Created" to the state "Running" when the execution of the first EDD Method in the Actions Proxy definition starts;
- in the meantime between the execution of two EDD Methods, the state machine remains in the state "Running";
- the state machine only transitions from the state "Running" to the state "Completed" when the execution of the last EDD Method in the Actions Proxy definition completes, or if any EDD Method exits for an unplanned reason.

All other state transitions remain the same.

5.12.4 Actions, EDD Actions and Actions Proxies

Actions are a provision of the FDI Server to allow FDI Clients to execute both EDD Methods in general and EDD Actions in particular. EDD Methods in general, with the exception of abort and action methods, are exposed in the Information Model as nodes under the ActionSet Object of the corresponding device or block node (see IEC 62769-5). EDD Actions on the other hand are not exposed in the Information Model. EDD Actions are made available to the FDI Client by putting the names of their corresponding Actions Proxies in the ListOfActions element in the XML description of the UID nodes (see 5.11.1).

The EDD Action types and the EDD constructs that use them are shown in Table 3 (see IEC 61804-3).

Table 3 – EDD Action types and the EDD constructs that use them

EDD Action type	EDD construct					UID
	VARIABLE	MENU	EDIT_DISPLAY	WAVEFORM	SOURCE	
Pre-read Actions	I	I				
Post-read Actions	I	I				
Pre-write Actions	I	I				
Post-write Actions	I	I				
Pre-edit Actions	E	E	E			E
Post-edit Actions	E	E	E			E
Init Actions		E		E	E	E
Exit Actions		E		E	E	E
Refresh Actions	I	E		E	E	E

As Table 3 indicates, in some cases, the FDI Server implicitly executes the EDD Actions (I), while, in other cases, the execution of EDD Actions is driven by the FDI Client, i.e. the FDI Client needs to explicitly start the execution of the EDD Actions in the FDI Server (E).

The FDI Server implicitly executes the following types of EDD Actions:

- Pre-read, post-read, pre-write and post-write actions, both for variables and menus;
- Refresh actions for variables.

Those types of actions shall not require user interaction; they are strictly intended to be used for Business Logic processing. Any action of one of those types that eventually requires user interaction will not perform the builtin but will return an error if possible.

The FDI Server implicitly handles pre-read, post-read and refresh actions for variables when the FDI Client reads online variables (see 5.7.3). Similarly, the FDI Server implicitly handles pre-write and post-write actions for variables when the FDI Client writes online variables, either in an immediate fashion (see 5.8.3) or in edit mode (see 5.8.4).

The pre-read, post-read, pre-write and post-write actions for menus are only used by the upload and download menus (see IEC 61804-4 and IEC 62769-2). The FDI Server implicitly handles pre-write and post-write actions for menus when the FDI Client transfers data to the device (5.2.2). Similarly, the FDI Server implicitly handles pre-read and post-read actions for menus when the FDI Client transfers data from the device (5.2.3).

The FDI Client explicitly starts the execution of the following types of EDD Actions:

- Pre-edit and post-edit actions for variables, menus, edit-displays and UIDs;
- Init, exit and refresh actions for menus, waveforms, sources and UIDs.

When those actions contain user interactions (see IEC 61804-4), they will require interaction between the FDI Server and the FDI Client. This is achieved by using Actions, as specified in 5.12.1. The FDI Client explicitly starts the execution of those types of actions in the FDI Server by calling the InvokeAction method and passing the name of the corresponding Actions Proxy as argument.

6 OPC UA services

6.1 OPC UA profiles

The set of services specified for OPC UA are grouped into standardized profiles as defined in IEC 62541-7. FDI Servers shall conform to the FDI Server Profile, which is specified as:

- including OPC UA "Standard Server";
- including OPC UA "DataAccess Server Facet";
- including OPC UA "Node Management Server Facet";
- including OPC UA "Method Server Facet";
- including OPC UA "Event Subscription Server Facet";
- including OPC UA "Auditing Server Facet";
- including FDI "FDI Information Model".

6.2 Service error information

6.2.1 Overview

FDI Servers provide service operations that are invoked through OPC UA services. Standard OPC UA service status information is returned by the FDI Server as a result of the service calls.

The OPC UA specification defines all services as having a standard response that includes a response header containing general and service-specific response codes in accordance with IEC 62541-4. The response code structure contains diagnostic information that returns both an error code as well as localized text for the error. FDI Servers shall fill in the diagnostic records including localized text for the reported errors.

The OPC UA diagnostic record allows Servers to include "inner" status information. FDI Servers will provide technology binding specific errors in the "inner" status record.

6.2.2 OPC UA services and their response

When the FDI Client submits a Service request Message to the FDI Server, if the service is supported and executed, the FDI Server generates a success/failure code that it includes in a positive response Message along with any data that is to be returned. Each Service request has a RequestHeader and each Service response has a ResponseHeader.

The ResponseHeader is a structure that has data members used to convey EDDL diagnostics information, the serviceResult and the diagnosticInfo.

The serviceResult is the standard, OPC UA-defined result of the Service invocation. The serviceResult type is StatusCode, which defines a standard numerical value that is used to report the outcome of an operation performed by an FDI Server. This code may have associated diagnostic information that describes the status in more detail.

The diagnosticInfo is a structure that is intended to return vendor-specific diagnostic information.

6.2.3 Mappings of EDDL response codes to OPC UA service response

When FDI Clients use OPC UA services to read and write the Attributes of Parameters, they receive back as part of the FDI Server response a ResponseHeader with success/failure code and diagnostics information.

The FDI Server uses the serviceResult and the diagnosticInfo data members of the ResponseHeader to return error and diagnostics information related to failure of execution of

EDDL variable actions, including PRE_READ, POST_READ, PRE_WRITE and POST_WRITE actions.

The StatusCode returned in the serviceResult data member of the ResponseHeader is also used to handle the EDDL VALIDITY attribute. Any attempt to access an invalid variable will be reported to the FDI Client as the result of a service call. The service returns a "Bad Failure" in the Severity bit of the StatusCode. In addition, the diagnosticInfo data member of the ResponseHeader can be used to provide detailed diagnostics on the failure. The FDI Server shall also deal with the fact that VALIDITY can be the result of the evaluation of a conditional expression. In that case, FDI Clients rely on the FDI Server notification capabilities when the model dynamically changes ~~due~~ owing to a conditional evaluation.

The serviceResult and the diagnosticInfo data members of the ResponseHeader are used to return error and diagnostics information related to EDDL response codes. EDDL response codes specify values that a device may return as the result of an operation. Each EDDL variable, record or value array can define its own associated set of response codes.

The serviceResult is used to return a status that corresponds to the EDDL response code TYPE attribute. The Severity bits of the StatusCode are set based on the response code TYPE according to Table 4.

Table 4 – OPC UA severity bits and EDDL response codes TYPE

OPC UA Severity	EDDL Response Codes Type
Good Success	SUCCESS
Uncertain Warning	MISC_WARNING, DATA_ENTRY_WARNING
Bad Failure	DATA_ENTRY_ERROR, MODE_ERROR, PROCESS_ERROR, MISC_ERROR

The symbolicIdIndex, localizedTextIndex and the additionalInfo data members of the diagnosticInfo are used to return the response code and the text description gotten from EDDL response codes definitions. It is the FDI Server's responsibility to translate the integer response code into its corresponding text description and fill in the diagnosticInfo.

The symbolicIdIndex data member is used to return the numeric response code from the EDDL RESPONSE_CODE. The numeric code shall be converted into a string in the stringTable. The symbolicIdIndex contains the index into the stringTable.

The localizedTextIndex data member is used to return the DESCRIPTION attribute from the EDDL RESPONSE_CODE. The DESCRIPTION string is conveyed to the FDI Client in the stringTable data member of the ResponseHeader parameter. The localizedTextIndex contains the index into the stringTable.

The additionalInfo data member is used to return the HELP attribute from the EDDL RESPONSE_CODE.

In addition to the response codes described via EDDL, standard response codes defined by the underlying communication protocols may also be returned.

6.3 Parameter value update during write service request

The FDI Server maintains an Information Model that contains Online variables that cache the value of the device variables. The specified behavior for OPC UA is for the Server to only store in the Online Variables those values that the Server has read from the device.

The FDI Server is not allowed, according to OPC UA specified behavior, to write a value both to the device and to the Online variable.

6.4 Localization

The Information Model defined for FDI, IEC 62769-5, is based on OPC UA. The OPC UA specification defines descriptive attributes and properties of elements to be localized strings. The FDI Server provides localized information for the OPC UA specified localized attributes and properties in the Device Type and Device Instance nodes.

The FDI Server uses information provided by the descriptive component in the FDI Package for a device type to create localized strings. FDI Packages support the specification of localized strings for descriptive information. If the device vendor provides such information, the FDI Server uses the appropriate localized string as the value for device type attributes and properties when responding to an FDI Client.

If the descriptive element in the FDI Package does not provide localized information, either no information or no information for the requested locale, the FDI Server will return the English default string.

Multiple clients connecting at the same time may eventually request the FDI Server to return localized attributes and properties in different languages. The FDI Server shall support multiple languages simultaneously when the clients requesting different languages connect to different Device Type or Device Instance nodes.

When clients requesting different languages connect at the same time to the same Device Type or Device Instance node, the support to multiple languages is optional. If the FDI Server does not support multiple languages in that situation, then it shall implement a "first wins" solution, i.e. it will use the language requested by the first client that connected to the Device Type or Device Instance node when returning localized attributes and properties to all subsequent clients.

6.5 Audit events

FDI Servers shall provide support for vendor-specific audit trail functionality. The support for auditing in the FDI Server is specified in IEC 62769-5.

7 Communication

7.1 Notation

Clause 7, describing communication, contains diagrams showing the Information Model, IEC 62769-5. The notation used in these figures uses the standards defined by OPC UA. These standards are summarized in IEC 62769-5.

7.2 General

7.2.1 Concepts

The FDI Server is responsible for managing communications. The FDI Server can support three types of communication infrastructure components:

- System communication hardware (see Figure 1);
- FDI Communication Server;
- Communication gateways (Nested Communication).

The FDI Server can support system-specific communication hardware (see Figure 1). The FDI Server interacts with the driver of the system communication hardware through proprietary interfaces to process fieldbus communication. System-specific communication management is not in the FDI specification's scope.

Figure 20 shows a possible architecture example of system communication integration.

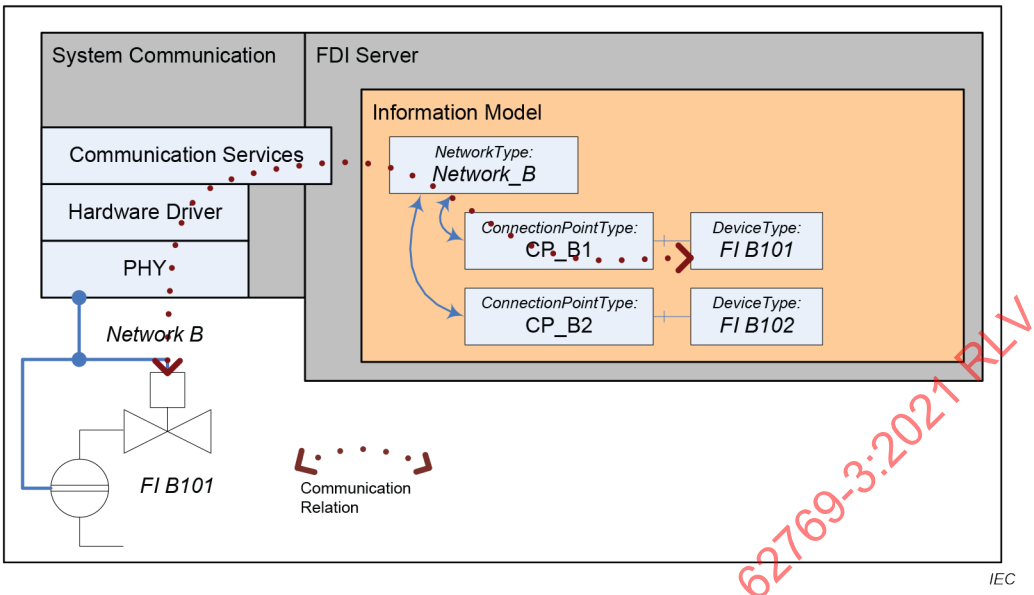

Figure 20 – System communication integration example

The FDI Server can implement access to physical networks through FDI Communication Servers (see Figure 1) (IEC 62769-7). The FDI Communication Server implements the access to the physical network. The interface between the FDI Server and the FDI Communication Server is based on OPC UA. The FDI Communication Server implements the OPC UA Server function. The FDI Server implements the OPC UA Client function. The FDI Communication Server implemented Information Model enables the access to the communication services.

An FDI Communication Server integration example is shown in Figure 21.

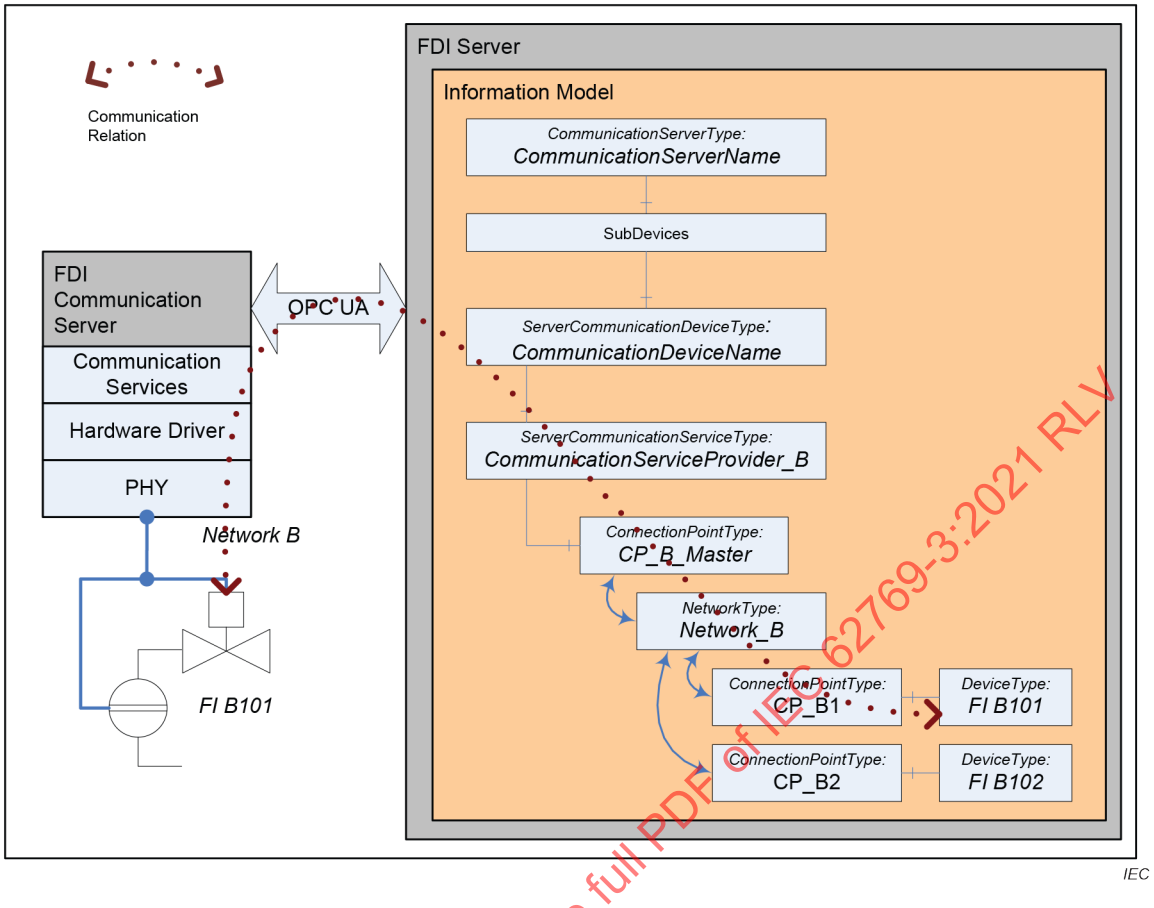

Figure 21 – FDI Communication Server integration example

In terms of the Information Model structure, the system communication hardware and the FDI Communication Server represent root communication devices.

FDI supports Nested Communication. The term "Nested Communication" stands for the ability of a system to process communication across protocol boundaries in heterogeneous networks. The insertion of additional communication gateway devices into the topology as shown in Figure 22 enables the handling of heterogeneous networks. These communication gateway devices implement the bridging functionally between different networks (gateway firmware). The gateway firmware implemented bridging functionality is also implemented in the Business Logic provided with the FDI Package describing the communication gateway. The FDI Server interacts only with this Business Logic of the communication gateway to process the Nested Communication function.

A communication gateway integration example is shown in Figure 22.

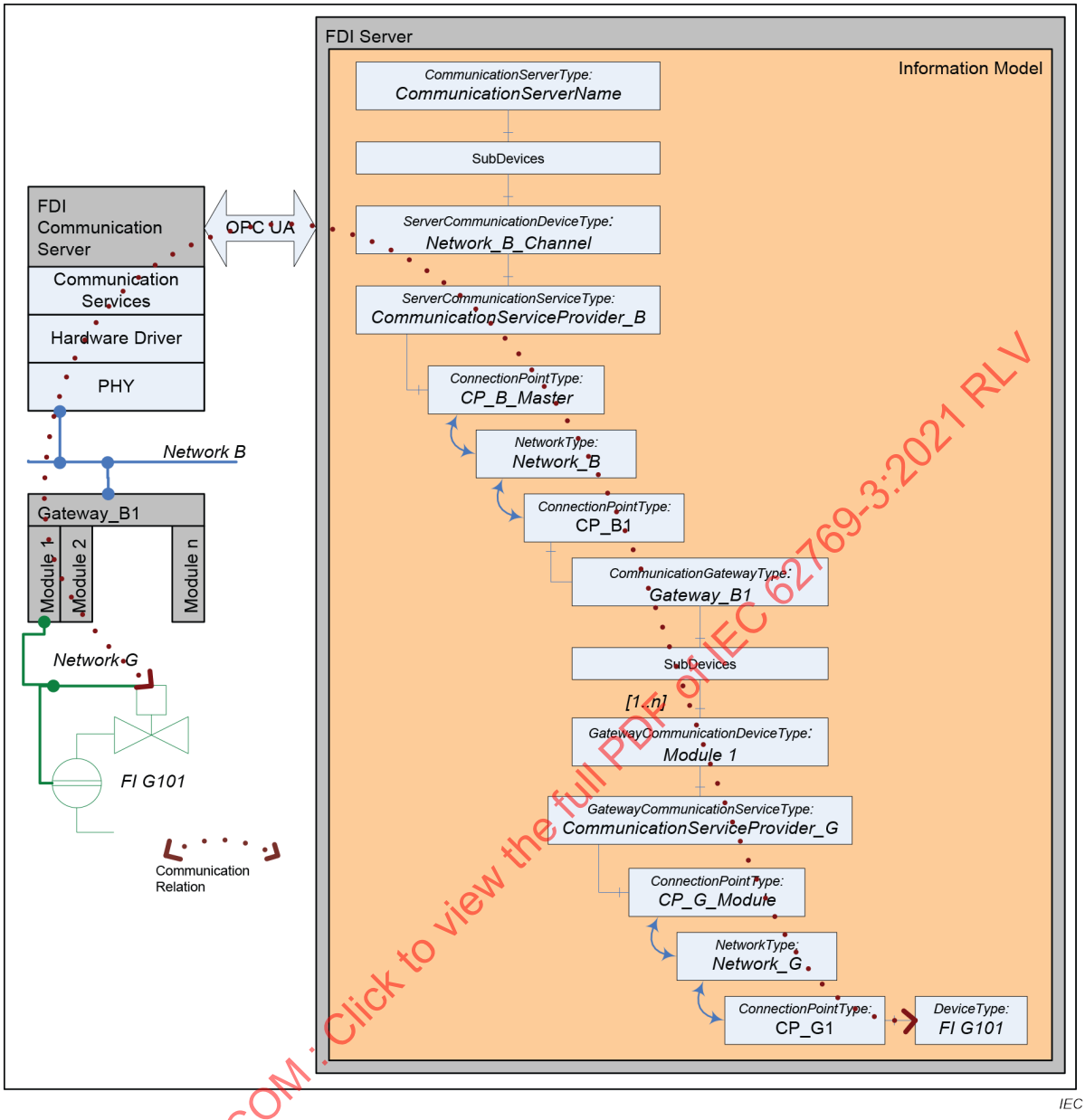

Figure 22 – Gateway integration example

7.2.2 Terms

The following contains a list of terms used in 7.3:

1) A Connection Point is an instance of a ConnectionPointType (see IEC 62769-5).
2) A Device is an instance of a DeviceType.
3) A Connection Point associated to a Device is called Device Connection Point.
4) A Connection Point associated to a Communication Device is called Communication Device Connection Point.
5) A Network is an instance of NetworkType (see IEC 62769-5).
6) FDI Communication Server (see IEC 62769-7).

7.3 Communication Service processing

7.3.1 Communication Service invocation

IEC 62769-7 specifies that both Gateways and FDI Communication Servers implement the communication services. Those services are specified in IEC 62769-7.

In order to allow the flexibility that is necessary to represent a variety of different scenarios involving communication between the communication server and the physical devices, IEC 62769-7 specifies that some communication services are provided through a communication service provider. While the communication service provider allows multitasking and enhances the communication capabilities of a communication device, it requires parallel execution in the server. The details about communication service providers (ServerCommunicationServiceType and GatewayCommunicationServiceType) are specified in IEC 62769-7. The requirements for parallel execution in the service are described through the rules stated in Clause 8.

The difference between a Gateway and an FDI Communication Server is about how or where these services shall be invoked:

a) If the communication device is an FDI Communication Server, the FDI Server invokes a communication service directly at the FDI Communication Server using an OPC UA Method service Call specified in IEC 62541-4. This call will end up in actual fieldbus communication.

b) If the communication device is part of a Gateway, the FDI Server invokes the communication service in terms of invoking an Action implemented by the Business Logic of the specific Gateway. The behavior (reaction) of the Gateway Business Logic is described in IEC 62769-7.

7.3.2 Analyze communication path

The Information Model defined in IEC 62769-5 supports a hierarchical topology. As shown in Figure 22, the topology reflects the physical network topology. The communication path analysis function allows the FDI Server to determine how communication messages need to be propagated from the Device that triggered a communication request to the Communication Device implementing the network access (root communication device) and which communication relations need to be activated before. Subsequent text will only consider the root communication device based on the FDI Communication Server.

The FDI Server identifies the communication path between a Device and an FDI Communication Server according to the following rules:

a) Topology iteration starts from the node representing the Device passing the elements Device Connection Point, Network, Communication Device Connection Point to the Communication Device within the same Network hierarchy. In this way, the FDI Server determines the local communication relation. A Communication Device that is associated next to the Device implements the communication service provider for this Device, this means the FDI Server shall propagate the communication service request between the Device and Communication Service Provider.

b) The FDI Server identifies a communication gateway along its Information Model structure as demonstrated in Figure 22. The key indicators are the Communication Device organized below a Device using the "has Component" relation. This specific device is called Gateway Head Station, which is connected to a different Network via a Connection Point. From here, the iteration continues as described in a).

c) The topology iteration procedure ends with finding the communication root device. The FDI Server identifies an FDI Communication Server (communication root device) because it has no association to other networks than the network for which the FDI Communication Server implements the communication service provider.

NOTE How the FDI Server determines System Communication Device is out of the scope of this document.

7.3.3 Manage communication relations

Prior to any data exchange related transfers, the FDI Server needs to establish or activate the communication relation between the Device representations in the Information Model and the physical network connected device. The invocation sequence of the communication service Connect on any of the Communication Devices along the communication path shall begin with the root communication device. The communication service Connect is specified in IEC 62769-7.

The Device Connection Point contains the address information to be used for the Connect service. The Information Model element FunctionalGroupType:Identification contains optionally required protocol-specific device type identification data. The Connect service argument names shall match with the browse names of the Information Model elements that hold related values (browse name matching).

The successful execution of service Connect activates the local communication relation between a Device and a Communication Device associated to the same network. The successful execution of service Connect on a network higher in a hierarchy is a prerequisite to a successful execution of the service Connect on a network lower in the hierarchy. The reason for this is the Gateway Business Logic that can invoke other communication services requiring an activated communication relation.

The FDI Server manages a CommunicationRelationId in accordance with IEC 62769-7.

A connection abort indication or the invocation of service Disconnect as described in IEC 62769-7 deactivates the local communication relation and any of the local communication relations in networks lower in the hierarchy.

7.3.4 Communication service request mapping

The FDI Server receives communication service requests from Devices or Gateways through:

a) the Online Variable Read;

b) the Online Variable Write;

c) the Business Logic invoking the communication related EDDL Builtin function, for example, send, send_all_values, send_command, send_command_trans, send_trans, send_value, WRITE_COMMAND, READ_COMMAND, and so on.

Like the Device, all of these communication service requests related source events apply to the EDDL PROFILE (IEC 61804-3). The FDI Server shall handle the communication service requests in accordance with EDDL-defined PROFILEs.

Since the EDDs shipped with the FDI Packages can describe relations between VARIABLE elements and COMMAND elements, the Variable Read or Write access can be mapped to a COMMAND description because of a particular COMMAND description that refers to a particular VARIABLE. Such COMMAND descriptions contain communication service arguments and instructions about how to create the data payload of a communication service.

If no COMMAND Description is present, the VARIABLE identifier (Name) and the VARIABLE value are the only communication service Transfer arguments.

Once the FDI Server has determined the communication service arguments from EDD, it can map it to the communication service Transfer (IEC 62769-7) arguments based on name matching. Transfer arguments shall have the same names, data types and semantics as described for a protocol-specific COMMAND definition.

EXAMPLE The COMMAND description contains the attributes SLOT, INDEX, REQUEST, REPLY. The protocol-specific signature of the Transfer service shall envision:

```
Transfer(
  [in]  String              communicationRelationId,
  [out] Integer             serviceError,
  [in]  unsigned char       SLOT,
  [in]  unsigned char       INDEX,
  [in]  char[]              REQUEST,
  [out] char[]              REPLY)
```

NOTE The arguments `communicationRelationId` and `serviceError` are described in IEC 62769-7.

7.3.5 Communication service request propagation

Subclause 7.3.5 describes how the FDI Server manages the communication message propagation along the communication path. The following Figure 23 represents an example scenario derived from Figure 22.

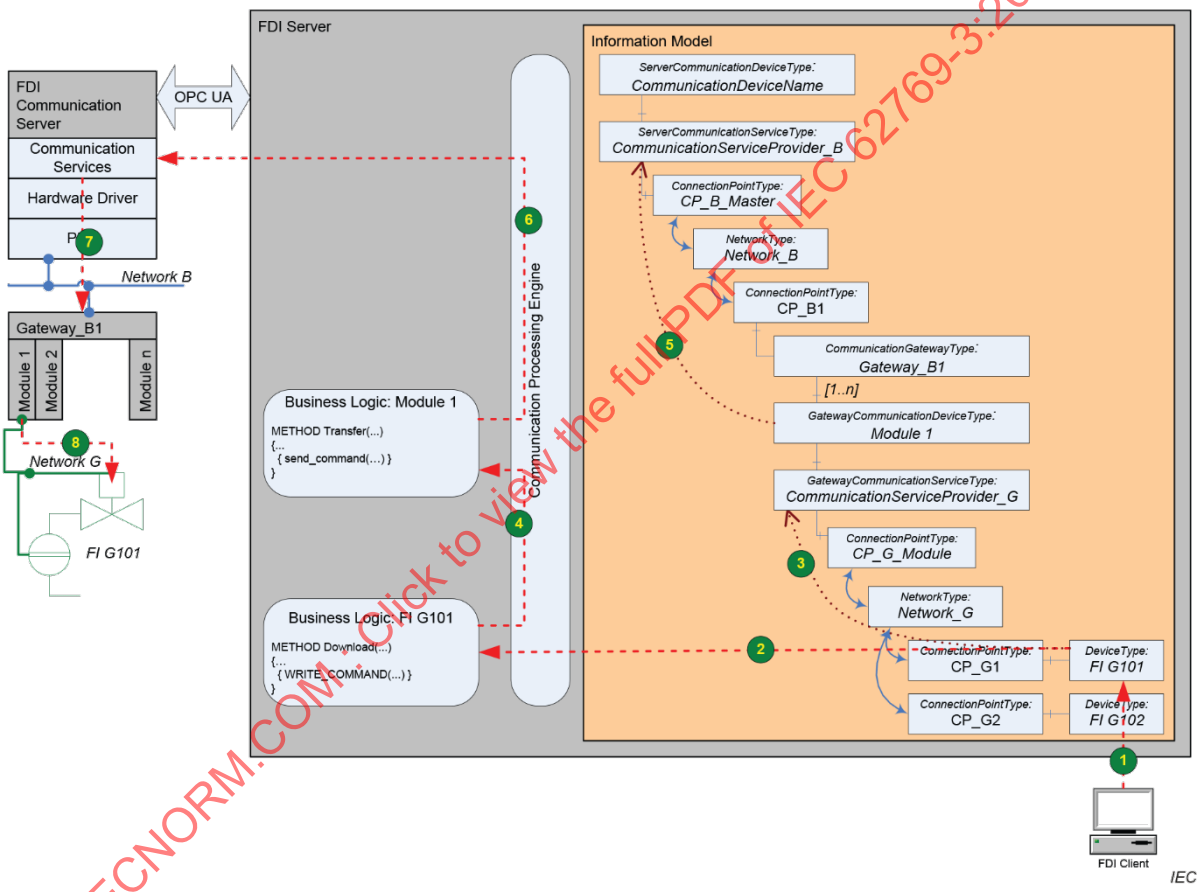

Figure 23 – Message propagation example scenario

NOTE 1 The numbers in brackets (1) to (8) used in the the following description refer to elements in Figure 23.

The FDI Server detects a communication request because of an FDI Client (1) invoking an Action (2). The processing of that Action (METHOD Download) invokes the communication request-related EDDL Builtin such as WRITE_COMMAND that has been mapped to Transfer service arguments, as described in 7.3.4.

The FDI Server processes the message propagation related iterations upwards through the hierarchy of the topology.

The FDI Server shall always determine the next Communication Service Provider along the communication path (3) and (5) (see 7.3.2) and invoke the service Transfer there.

If the Communication Service Provider processing the Transfer service is in an FDI Communication Server, the Transfer service needs to be invoked using the OPC UA service Call (6).

NOTE 2 The Communication Service Provider in the FDI Communication Server sends the protocol-specific message to the physical network (7).

If the Communication Service Provider processing the Transfer service is in a Gateway, the processing of that related Business Logic will cause other communication request-related EDDL Builtin invocations, for example, send_command (4). The iteration procedure enters the next recursion determining the next Communication Service Provider along the communication path (5) (see 7.3.2) and invokes the service Transfer there.

NOTE 3 The gateway implementation of service Transfer is device-specific. The Transfer logic wraps the incoming Transfer argument values and creates another message to be sent out calling a communication request related EDDL Builtin (see IEC 61804-5).

The Transfer logic can invoke multiple communication requests as this might be needed to manage the protocol bridge function. The physical gateway device unwraps and forwards the message (4) to the physical device (8).

The FDI Server managed communication propagation process is a recursive process in which the Business Logic execution of one Device can invoke the Business Logic execution of a different Device. This FDI Server needs to maintain an invocation stack.

7.3.6 Communication error handling

The FDI Server is responsible for handling communication errors. The FDI Server detects errors either from the Communication Service Provider returned service invocation results as specified in IEC 62769-7 or through EDDL builtins for abort processing.

The FDI Server aborts all communication Actions waiting for a response if a communication error or abort is received.

The FDI Server will return a failure to the originating service if a communication error or abort is received.

7.4 FDI Communication Server specific handling

7.4.1 Discovery

IEC 62769-7 describes the FDI Communication Server implemented discovery support in terms of:

a) VARIABLE definitions describing the FDI Communication Server's identification data that are represented in the FDI Server hosted Information Model;

b) FDI Communication Server implemented usage of IEC 62541-4 specified discovery services.

The FDI Server uses the services FindServers and the GetEndpoint IEC 62541-4 specified discovery service set to determine the FDI Communication Server. The FDI Server shall match the FDI Communication Server's defined identification data with values returned from the functions FindServers and GetEndpoints.

The FDI Server implements the IEC 62541-4 specified Discovery Server.

7.4.2 Information Model synchronization

In accordance with IEC 62769-7, the FDI Communication Package contains an EDD element describing the VARIABLES and Business Logic mapped into the Information Model. IEC 62769-7 also describes the overlap between the FDI Communication Server-hosted Information Model and the FDI Server-hosted Information Model. This overlap represents the shared Information Model.

The FDI Server synchronizes the shared Information Model:

a) Any access to an offline node of the Information Model is locally handled through the Information Model.

b) The FDI Server handles a write access to an online node of the Information Model by performing the same write access in FDI Communication Server-hosted Information Model.

c) Any read of an online node of the Information Model results in a read operation on the corresponding node of the FDI Communication Server-hosted Information Model.

d) Any configuration changes affecting the modular structure represented in the Information Model are copied from the FDI Server-hosted Information Model to the FDI Communication Server-hosted Information Model.

8 Parallel Execution within the FDI Server

8.1 Motivation

Within the EDDL concept, each device is described by a set of parameters and an EDD that describes and handles relations between the parameters and their attributes. Each combination of the EDD and the respective set of parameters builds an entity describing a device instance (called EDD Entity hereafter). EDDL allows only synchronous operation with such an entity without any parallel execution. Therefore, when the EDD Entity is performing an action (e.g. reading a variable value, executing a method, etc.), any other action request shall be postponed until the action execution is finished.

As long as there is no relation between EDD Entities the action for EDD Entities can be executed independently and subsequent action requests can be queued without any risk to force a deadlock.

As soon as relations between EDD Entities have to be handled, the FDI Server has to control the execution within the EDD Entities in a way that deadlock scenarios or parallel execution within one EDD Entity are prohibited.

Nested communication is a concept based on interaction between EDD Entities. When handling multiple communications at once, the FDI Server has to handle actions in the FDI system in parallel. To prevent deadlock scenarios, the FDI Server has to follow well-defined execution rules.

8.2 Internal structure of the EDD interpreter

A core component of an FDI Server is the EDD interpreter (see Clause A.1). The EDD interpreter can be seen as a component that consists of EDD entities. Each EDD entity itself consists of the following parts:

- An associated EDD for a device or a component of a modular device.
- A set of data representing the state of the EDD entity and containing data that the interpreter requires to run the EDD associated with the data set. This data for example might contain the offline data set for the associated device and cached data of the connected device. It also contains additional information the interpreter needs for EDD-specific calculations.
- The interpreter logic that is triggered from outside and when triggered interprets the EDD, performing subsequent activities, changes the state and delivers calculation results.

8.3 Rules for running an EDD entity

As mentioned in 8.2, an activity at EDD entities is initiated always by a trigger. There are two kinds of triggers:

a) A trigger from the EDD entity itself that the interpreter logic requires for a correct EDD execution. An example for such triggers is periodic updates of dynamic variables.

b) A trigger that is a consequence of a service request from outside the FDI Server.

   For example:

   1) service requests from an FDI Client;

   2) service requests from an OPC-UA client.

The execution of an activity at an EDD entity ~~has to~~ shall follow the rules given below:

c) An activity at an EDD entity is always a consequence of a trigger.

d) An activity at an EDD entity cannot be interrupted.

e) An activity runs until the activity is finished or aborted.

f) An EDD entity can only execute one activity at a time.

g) An activity executed by an EDD entity always performs a non-interruptible process from the perspective of the EDDL logic. Such processes are for example:

   1) calculation of EDD objects,

   2) reading a variable from a device,

   3) writing a variable to a device,

   4) editing a variable,

   5) any activity that is embraced with pre- and post actions,

   6) executing an EDD method.

h) An active EDD entity may initiate sub-activities. While a sub-activity is ongoing the current activity at the EDD entity is paused. There are two kinds of sub-activities:

   1) the active EDD entity calls another EDD entity (e.g. nested communication or calls using cross-block and cross-module references),

   2) the active EDD entity requests another external service (e.g. communication, request for user interaction).

i) While an activity is paused re-entrance for activities is possible for those activities that are a consequence of the paused activity. Activities started by other triggers have to be blocked until the paused activity at the entity is finished.

j) When an activity chain of a trigger is blocked, there exists a blocking relation to another trigger and its activity chain. If this activity chain is blocked too, there is again a blocking relation to another trigger. Such series of blocking relations have to be monitored for recursion each time an activity chain has to be blocked. Recursion in the series of blocking dependencies indicates a deadlock scenario.

k) The server can resolve deadlock scenarios by aborting one of the involved activity chains.

For illustration, some examples on how to run an EDD entity in the FDI Server are described in Annex C.

Annex A
(informative)

FDI Server functional structure

A.1 FDI functional elements

The normative definition of an FDI Server is as shown in Figure 1. A non-normative view of an FDI Server shows the functional components that comprise the FDI Server and is shown in Figure A.1.

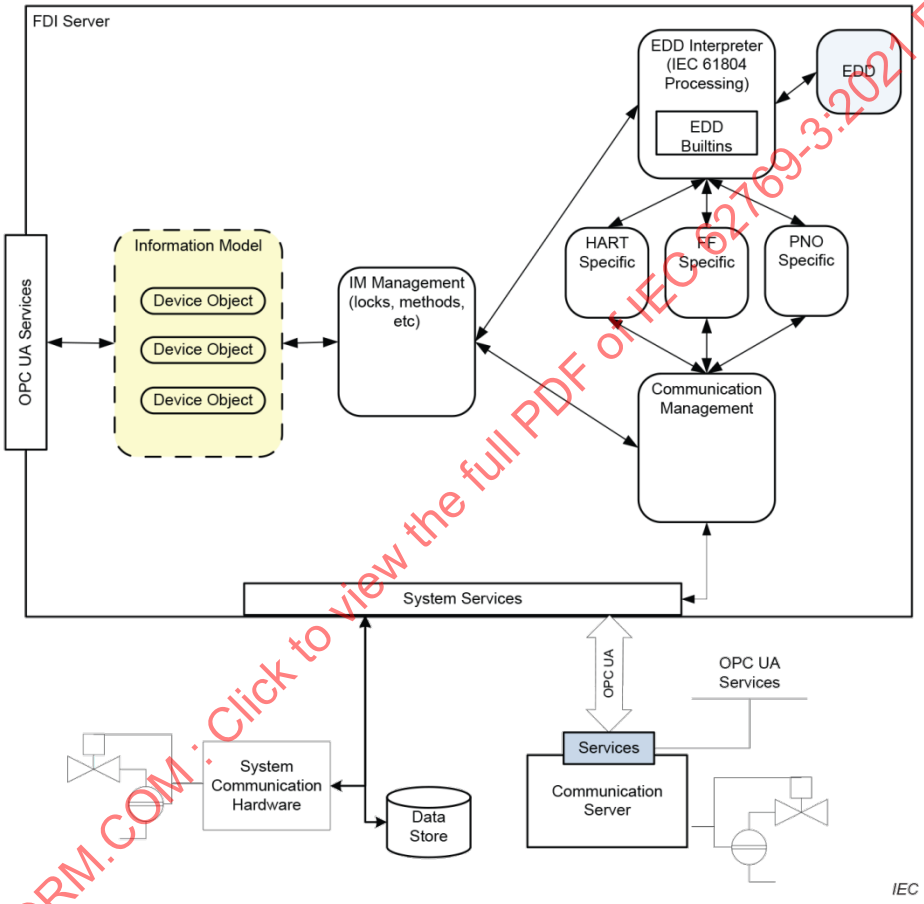

Figure A.1 – Functional components of an FDI Server

The FDI Server functionally contains an EDD interpreter that conforms to IEC 61804-3. The EDD interpreter provides descriptive information that is exposed in the Information Model, for example, device variables and standard menus. The EDD interpreter also provides the method execution functionality for IEC 61804-3.

Although IEC 61804-3 defines a standard EDD language, there are protocol-specific differences between EDDs. The FDI Server implements protocol-specific components. The protocol-specific components are used both as part of the EDD interpretation as well as for formatting communication.

The FDI Server contains node management functionality that provides the support for the Information Model. The Information Management component maintains the Information Model, handles multi-user requests, including lock management, and executes methods. The functionality provided by the Information Model management component is not restricted to FDI functionality. The IM management component may also provide general OPC UA functionality unrelated to FDI.

Service requests that result in physical read and write operations are passed to a communication manager. The communication manager provides the functionality required for communication, including state management of the communication requests. The communication manager contains the information to interact with system communication devices.

The communication manager interacts with protocol-specific components to create the actual messages transmitted on the fieldbus. The protocol-specific components interact with the EDD interpreter to retrieve information from the EDD to create the protocol-specific messages. The messages may be commands as in HART or the messages may be service requests as in FF. The actual message is created by the protocol-specific component.

The communication manager is responsible for managing the Nested Communication. The communication manager initiates the communication chain through the creation of protocol-specific messages. The communication manager then passes the message through the chain of communication devices in the topology until a top-level device is reached. The communication manager then interacts with the communication drivers for the top-level device. The interaction may be proprietary if the communication is through a system device. The communication may also be standardized through OPC UA to an FDI Communication Server.

A.2 FDI Server extension

An FDI Server can be extended to support future descriptive and protocol technologies through the addition of new interpreters and protocol handlers as illustrated in Figure A.2.

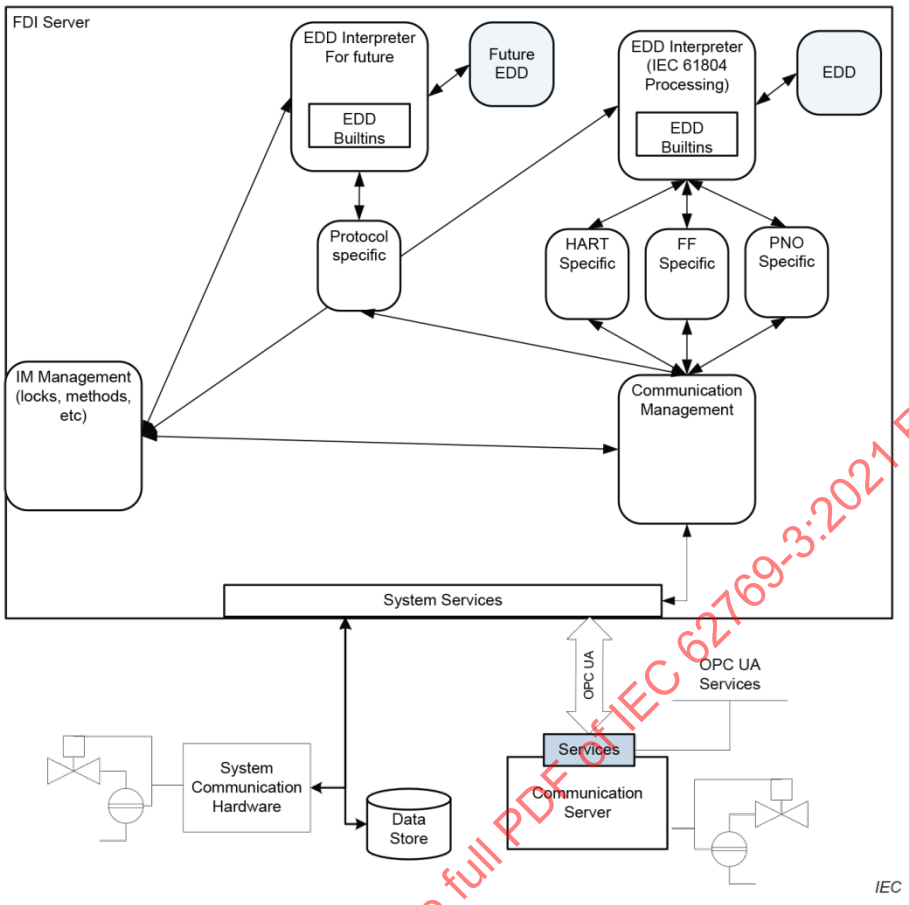

Figure A.2 – FDI Server extensions

These extensions, just like the support for FF, HART, and PNO, are specific to the implementation provided by the vendor of the FDI Server.

Annex B
(informative)

Access privileges and user roles

B.1 User roles and usage case

The specification for the Device Definition of an FDI Package contains a CLASS attribute in some elements. The CLASS attribute is supplied by the device vendor in the FDI Package and defines the intended usage of the Information Model element. The FDI Server makes the usage categories available to the system so user access and visibility to Information Model elements can be controlled by the system, possibly through the enforcement of system policies. The FDI Server reacts to the system defined policies to independently enforce the read/write access and public/private visibility of Information Model elements made available to users.

Figure B.1 depicts the relationship between the FDI Package, FDI Server, FDI Client, system, and user.

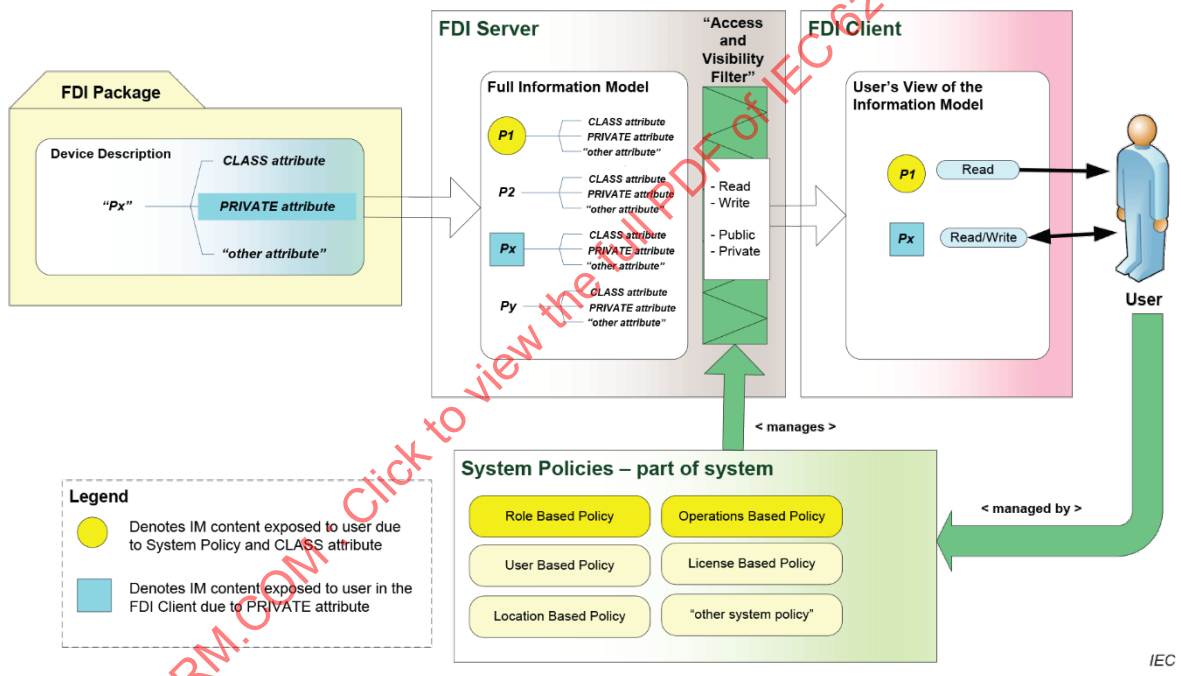

Figure B.1 – User roles and access privileges

The following relationships are noted:

The FDI Device Package identifies attributes associated with each element in the device description. These attributes become part of the Information Model that is managed by the FDI Server.

The CLASS attribute is made available to the system through the Information Model. It is an attribute for identifying the use cases, or usage scenarios, that are applicable to the Information Model element. The FDI host can use the CLASS attribute – in particular the value SPECIALIST – to determine whether the user of the FDI Client has access to the Information Model element and what level of access is allowed. The mechanism for making this determination is part of the system policy model that is internal to the system and outside of the scope of the FDI.

The system shall convey the access level for each Information Model element to the FDI Server so it can enforce the access model of the Information Model element for the user. The FDI Server is only the enforcer of the access rules, it does not decide "who" or "why". The system makes all of the "who" and "why" decisions, using both the CLASS attribute provided in the FDI Package and the system policies managed by the system. Some potential system policies may include, but are not limited to the following:

- Role-Based Policy,
- User-Based Policy,
- Location-Based Policy,
- Operations-Based Policy,
- License-Based Policy,
- Other Policies.

B.2 Private data usage

Among the attributes that have been prescribed by the FDI specification is an attribute for identifying whether an Information Model element, including data and Actions, is private to the elements in the FDI Device Package. This attribute, referred to as PRIVATE attribute in Figure B.1, determines whether an element in the Information Model is visible to FDI Clients during browse operations on the Information Model. The FDI Package elements have prior knowledge of private data and Actions in the Information Model and are able to access these private elements in accordance with access rules defined by CLASS attribute and system policy.

Access and visibility are independent attributes. For example, a private Action may be limited to user access during online usage scenarios.

System policy shall not be allowed to override the PRIVATE attribute in the Information Model. For example, the system policy cannot make private data public or public data private. The PRIVATE attribute is internally managed by device vendors.

Annex C
(informative)

Parallel execution within the FDI Server – Examples

C.1 Simple example for a synchronous execution

The generic examples in Clause C.1 are intended to visualize, for a better clarification, the rules given in 8.3.

Figure C.1 schematically shows the simplest example of a synchronous execution of two triggered activities. Both activities can be executed independently, because different EDD entities are involved.

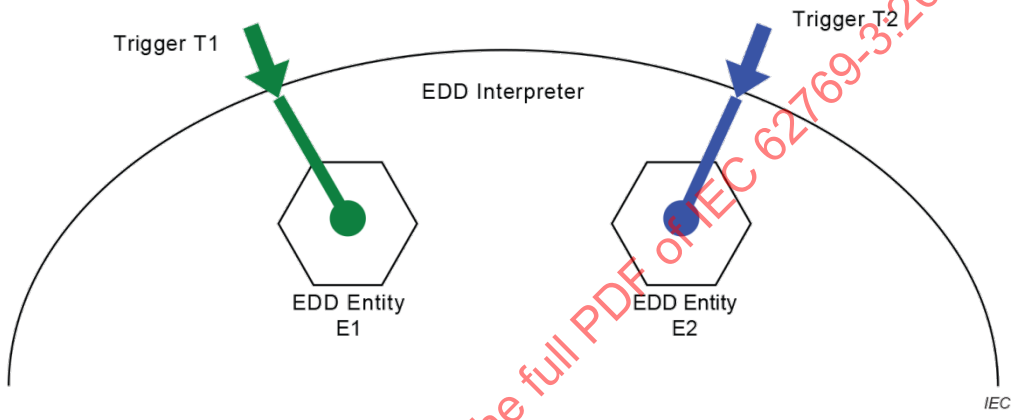

Figure C.1 – Synchronous execution of two triggers

C.2 Example for a concurrent execution

Figure C.2 shows a use case where two synchronous triggers try to access one and the same activity. While the activity of trigger T1 is executed, trigger T2 is blocked.

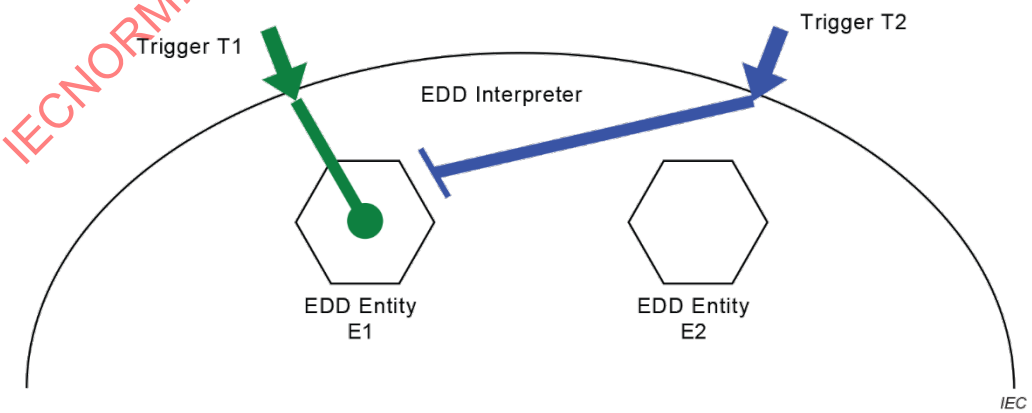

Figure C.2 – Concurrent execution of two triggers (step 1)

In Figure C.3 the activity of T1 is paused in EDD Entity E1 to execute a sub activity in E2. The activity of T2 is blocked until the activity of T1 is finished.

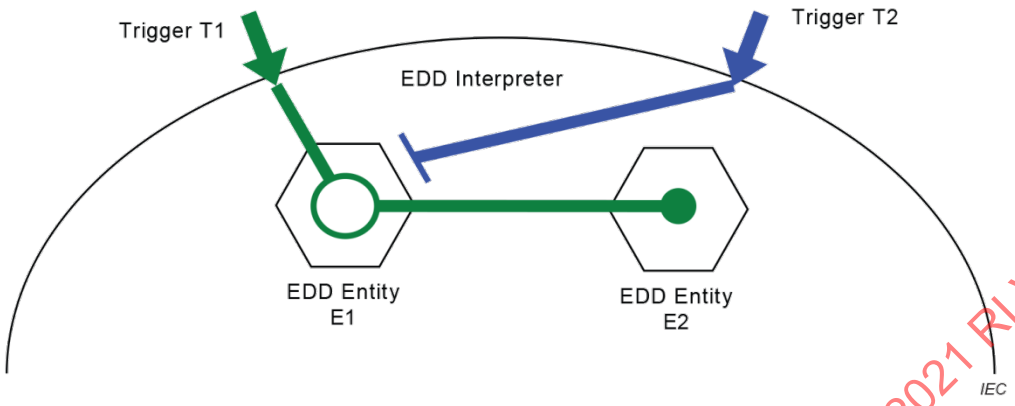

Figure C.3 – Concurrent execution of two triggers (step 2)

In Figure C.4, the sub activity in E2 initiates another sub activity back again in E1. This call back is allowed, because it is a direct consequence within the activity chain initiated by trigger T1 while activities of trigger T2 continue to be blocked.

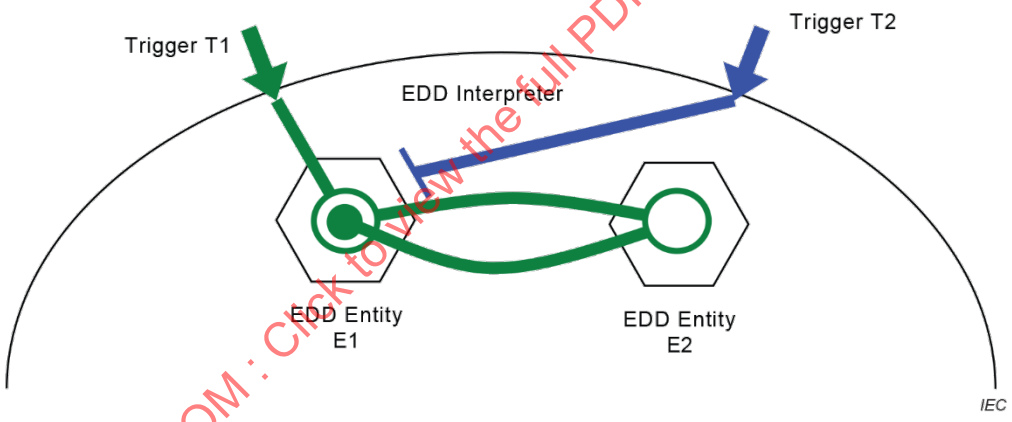

Figure C.4 – Concurrent execution of two triggers (step 3)

Figure C.5 shows that the activity of trigger T1 is finished in EDD Entity E1. Now the activity of trigger T2 can be started.

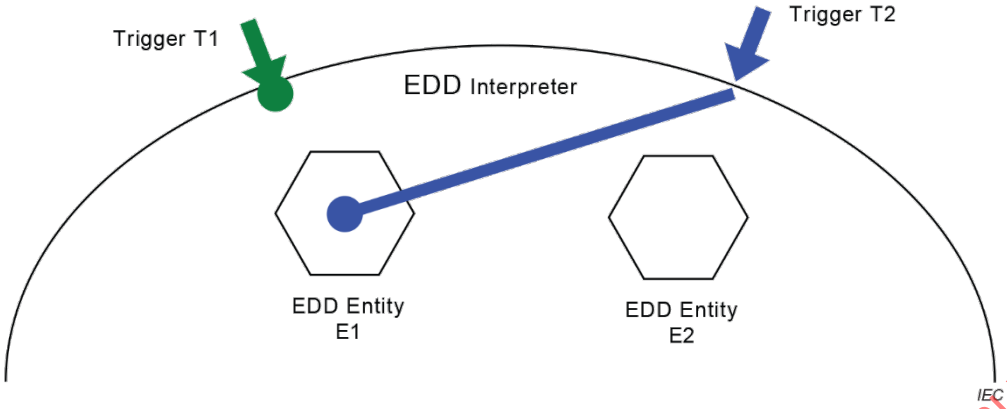

Figure C.5 – Concurrent execution of two triggers (step 4)

C.3 Deadlock detection in concurrent execution

A deadlock situation is shown in Figure C.6. The activity of trigger T1 wants to access EDD Entity E2 as a sub activity of an activity in EDD Entity E1 and trigger T2 wants to access E1 as a sub activity of an activity in E2 at the same time. Both activity chains deadlock one another.

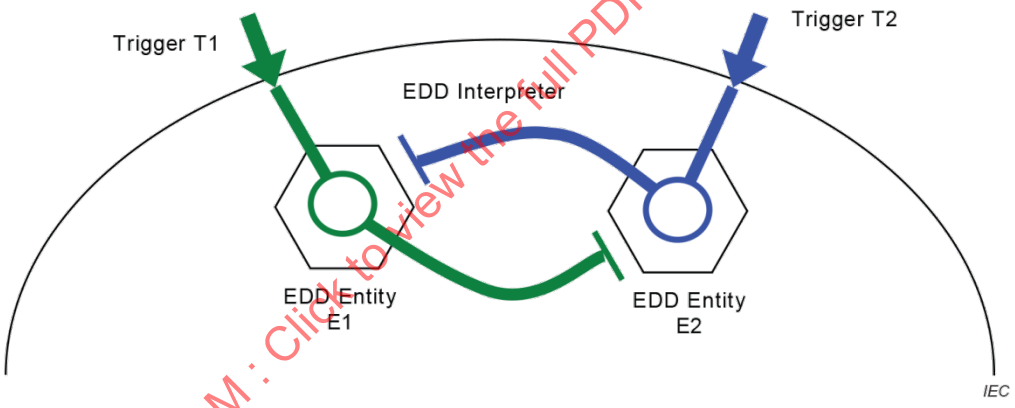

Figure C.6 – Concurrent execution of two triggers

In fact, the activity chain of trigger T1 is blocked by the activity chain of trigger T2 and vice versa. The above example is one of the simplest deadlock scenarios, probably it will happen that much more complex deadlock scenarios occur that usually have a couple of more activity chains involved.

Independently from the complexity of a deadlock scenario, there exists a simple rule to detect deadlocks by monitoring blocking dependencies of activity chains. If activity chain of trigger T(n) is blocked by activity chain of trigger T(n + 1), and T(n + 1) is blocked by T(n + 2), a deadlock scenario is reached when this relation circles back and a trigger T(m) is blocked by T(n). It is not a deadlock scenario as long as the series of blocking dependencies ends up in a non-blocked activity chain.

Usually, it can be expected that the reason for a deadlock scenario is found in an involved device package. Therefore, device package developers should use cross-block and cross-module references only with care and caution.

Nevertheless, the FDI Server is responsible for detecting deadlock scenarios. If the FDI Server has detected a deadlock scenario, it can break it by aborting one of the involved activity chains and even recall the aborted trigger at a later time.

_____

IEC 62769-3

Edition 2.0 2021-02

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Field device integration (FDI) – Part 3: Server

Intégration des appareils de terrain (FDI) – Partie 3: Serveur

IEC 62769-3:2021-02(en-fr)

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

FIELD DEVICE INTEGRATION (FDI) –

Part 3: Server

FOREWORD

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

International Standard IEC 62769-3 has been prepared by subcommittee 65E: Devices and integration in enterprise systems, of IEC technical committee 65: Industrial-process measurement, control and automation.

This second edition cancels and replaces the first edition published in 2015. This edition constitutes a technical revision.

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

a) modification of the edit context concept to harmonize the IEC 61804 and the IEC 62769 series.

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

FDIS	Report on voting
65E/760/FDIS	65E/770/RVD

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

This document 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 document will remain unchanged until the stability date indicated on the IEC website under "http://webstore.iec.ch" in the data related to the specific document. At this date, the document 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 IEC 62769 series has the general title *Field Device Integration (FDI)* and the following parts:

– Part 1: Overview
– Part 2: FDI Client
– Part 3: FDI Server
– Part 4: FDI Packages
– Part 5: FDI Information Model
– Part 6: FDI Technology Mapping
– Part 7: FDI Communication Devices
– Part 100: Profiles – Generic Protocol Extensions
– Part 101-1: Profiles – Foundation Fieldbus H1
– Part 101-2: Profiles – Foundation Fieldbus HSE
– Part 103-1: Profiles – PROFIBUS
– Part 103-4: Profiles – PROFINET
– Part 109-1: Profiles – HART and WirelessHART
– Part 115-2: Profiles – Protocol-specific Definitions for Modbus RTU
– Part 150-1: Profiles – ISA 100.11a

FIELD DEVICE INTEGRATION (FDI) –

Part 3: Server

1 Scope

This part of IEC 62769 specifies the FDI Server. 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 figure.

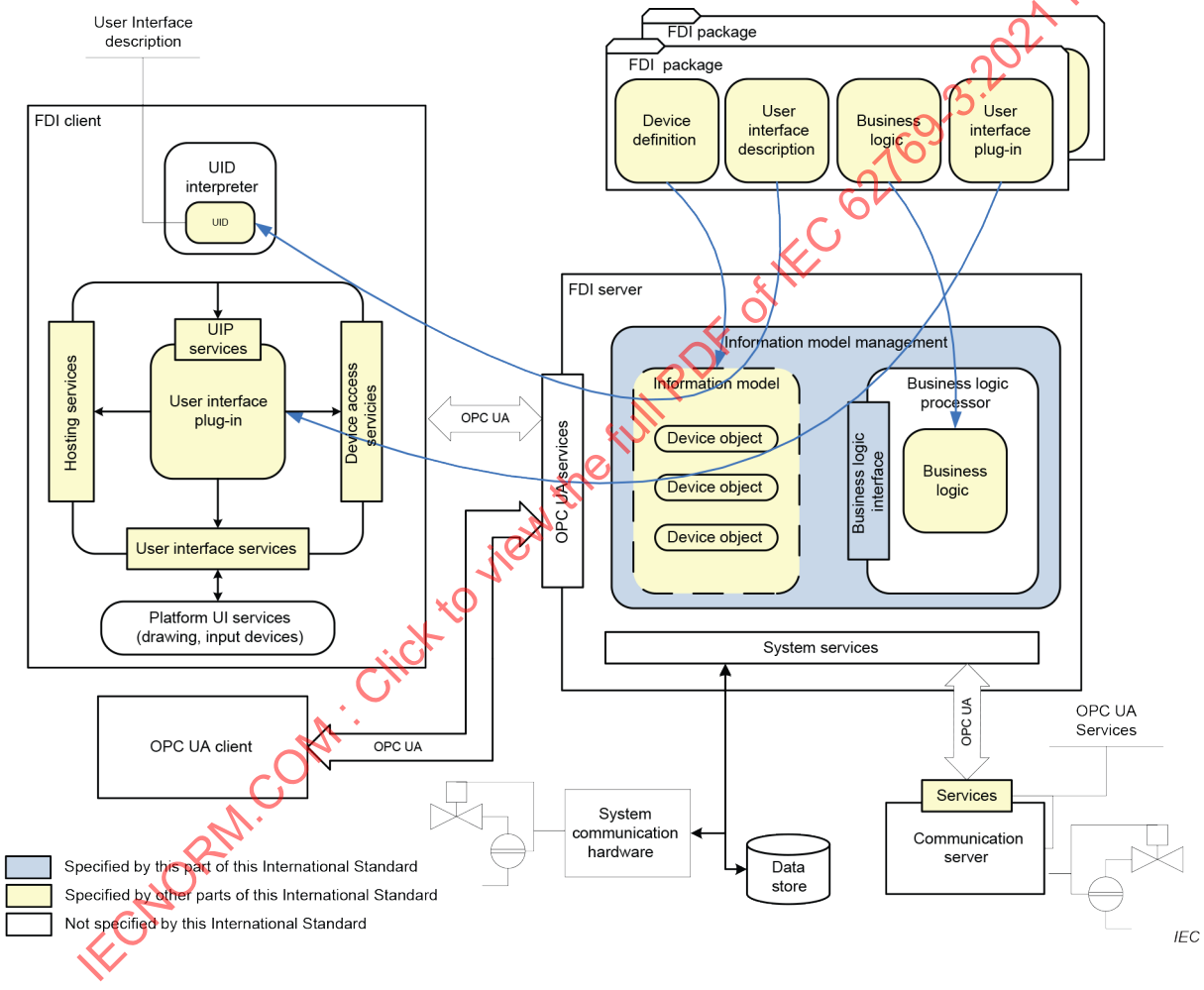

Figure 1 – FDI architecture diagram

2 Normative references

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

IEC 61804 (all parts), *Function blocks (FB) for process control and electronic device description language (EDDL)*

IEC 61804-4:2020, *Function blocks (FB) for process control and electronic device description language (EDDL) – Part 4: EDD interpretation*

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

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

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

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

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

IEC 62769-5, *Field Device Integration (FDI) – Part 5: FDI Information Model*

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

3 Terms, definitions, abbreviated terms and conventions

3.1 Terms and definitions

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

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

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

3.1.1
Actions Proxy
internal FDI Server entity that encapsulates all the EDD Methods specified in an EDD Action definition

3.2 Abbreviated terms

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

3.3 Conventions

For the purposes of this document, the conventions given in IEC 62769-1 apply.

4 Overview

The structure for an FDI Server is shown in Figure 1.

FDI Servers that support connectivity with third-party FDI Clients shall support OPC UA. A vendor can provide both an FDI Server and one or more FDI Clients. In this case, the FDI Clients can communicate with the FDI Server through proprietary protocols.

An FDI Server communicates with devices via Native Communication (see 7.2.1) and/or Communication Devices (see IEC 62769-7).

An FDI Server provides information to FDI Clients through an Information Model (see IEC 62769-5) as follows.

- The Information Model includes information about Device Types and Device Instances. The information for a Device Instance includes offline data (engineering data), as well as online data (values from the physical device).
- The Information Model is created using information from FDI Packages. However, not all of the information in an FDI Package is reflected in the Information Model.
- Referential integrity of the Information Model is maintained using information from FDI Packages.
- FDI Packages can contain Attachments that contain device manuals and protocol-specific information (see IEC 62769-4). Those Attachments, including device manuals and protocol-specific support files, are exposed via the Information Model.
- FDI Device Packages contain information about device types (see IEC 62769-4). Each device type defined in a package is mapped to a distinct DeviceType node in the Information Model.
- FDI Profile Packages are used to provide interaction with devices for which an FDI Device Package does not exist (see IEC 62769-4).
- Multiple revisions of an FDI Package generate distinct DeviceType nodes in the Information Model (see IEC 62769-4).

FDI Packages contain digital signatures that allow an FDI Server to authenticate their contents (see IEC 62769-4). An FDI Server shall not use an FDI Package if the digital signature provided by the FDI Package is invalid.

An FDI Server shall verify the FDI Technology Version (see IEC 62769-1) of any FDI Package it uses to ensure the FDI Package is compatible with the FDI Server.

The resulting functional structure of an FDI server is described in Annex A.

5 Information Model

5.1 General

The FDI Server shall use the Device Definition of an FDI Package to maintain the Information Model.

The Device Definition can contain conditional expressions. Conditional expressions are used when a certain aspect of the Device Definition is not static but rather is dependent on the state of the device. Whenever the online or offline values of a Device Instance are modified, the FDI Server shall re-evaluate the relevant conditional expressions and modify the Information Model accordingly.

The evaluation of conditional expressions can invalidate variables in the Information Model. The FDI Server shall change the AccessLevel attribute of invalidated variables such that they are neither readable nor writable and the status of these variables shall be set to bad. Read and write service requests for invalidated variables shall return a failure.

The Device Definition can specify relationships between variables in a device. These relationships can impact the value of variables in the Information Model.

The FDI Server shall generate DataChange Notifications to any FDI Clients that are subscribing to Information Model elements that have changed.

FDI Packages provide Business Logic that is used by the FDI Server to maintain the integrity of the Information Model. The Business Logic specified in an FDI Package can invoke builtin functions that shall be implemented by the FDI Server. The builtin functions that shall be implemented by the FDI Server are specified in IEC 61804-5.

5.2 Online/Offline

5.2.1 Overview

The Information Model maintained by the FDI Server contains online and offline values. The online values reflect values in a physical component/device. The offline values reflect values stored in a configuration database.

The offline values are updated through write service requests from an FDI Client or Business Logic executed by the FDI Server. The offline values are not updated when the FDI Server reads data from the device or writes data to the device.

The online values in the Information Model are not updated through write service requests. Successful write service requests through the Information Model result in value changes in the physical devices. The online values in the Information Model will then be updated as a result of read service requests or subscriptions.

FDI Servers can provide a server-specific mechanism for creating Device Instances without the presence of physical hardware. The FDI Server creates these instances using information in FDI Packages. All read/write requests for online values for Device Instances with no physical device shall return an error.

The transfer of information between the offline values and the physical device is supported through the TransferToDevice and TransferFromDevice methods in the Information Model. These Methods shall implement the download and upload procedures, respectively, as specified in IEC 61804-4. When no implementation is provided based on IEC 61804-4, then these Methods shall return Bad_NotSupported, as per IEC 62541-4.

The Device shall have been locked prior to invoking these methods, as specified in IEC 62769-5.

5.2.2 Transfer to device

The TransferToDevice method shall implement the download procedure as specified in IEC 61804-4. This transfers the offline values to the physical device.

As a general rule, the FDI Server should not change the Online variable node when writing a value to the device. The Online variable node should be updated only in the process of read operations or subscriptions. Notwithstanding, as specified in IEC 62769-5, the FDI Server will reset any cached Value for the target Nodes in the Information Model so that they will be re-read next time they are requested.

The status information returned for each variable included in the write service request is used to compose the TransferResult, as specified in IEC 62769-5.

5.2.3 Transfer from device

The TransferFromDevice method shall implement the upload procedure as specified in IEC 61804-4. This transfers the values from the physical device to the offline values.

If any read operations from the device fail during upload, the corresponding offline value shall not be modified.

The status information returned for each variable included in the read service request is used to compose the TransferResult, as specified in IEC 62769-5.

5.3 Access privileges

Systems implement security and access policies based on a number of characteristics such as user role and plant area. FDI Servers use these policies, along with information in FDI Packages, to determine the access privileges granted to the user.

The elements of an FDI Package can be associated with one or more usage attributes. The FDI Server uses these attributes to set the UserAccessLevel attribute of Variables and the UserExecutable attribute of Methods. The usage attributes in an FDI Package are simply hints to be used by the FDI Server, i.e., they may be disregarded or overridden by the FDI Server. See also Annex B.

5.4 Private Parameters

The Parameters and Actions specified in an FDI Package may be declared private. Private Parameters and Actions shall not be browsable; they shall only be accessible through references from other elements of an FDI Package.

More specifically, the FDI Server shall support private Parameters and Actions as follows.

- The FDI Server shall create nodes in the Information Model for the private Parameters and Actions.
- The FDI Server shall not include information about private Parameters and Actions in a response to a Browse, BrowseNext, QueryFirst, or QueryNext service request.
- The FDI Server shall return the NodeIds of private Parameters and Actions when the name of a private Parameter or Action is passed to TranslateBrowsePathsToNodeIds.
- The FDI Server shall process a read/write service request for a private Parameter in the same way as it does for public (browsable) Parameters (see 5.7 and 5.8).
- The FDI Server shall execute private Actions in the same way as it does public (browsable) Actions (see 5.12).

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.

5.5 Locking

The FDI Server provides locking services to grant FDI Clients exclusive access to Device and Network elements in the Information Model. The locking services consist of a set of Methods and status information. The methods, and their behavior, are specified in IEC 62769-5.

The following behavior shall be implemented by the FDI Server to support locks.

- Locking applies to both online and offline nodes.
- Once locked by one FDI Client, any attempt to write to a Parameter or to execute an Action by another FDI Client shall be rejected.
- Locking is not required for read services.
- Parameters that are locked by one FDI Client can still be read by other FDI Clients, i.e., read requests on a Parameter that is locked are not rejected.

Internal use of the locking mechanism for maintaining the Information Model integrity is FDI Server vendor-specific.

Figure 2 illustrates a locking sequence with multiple service invocations during the locked state.

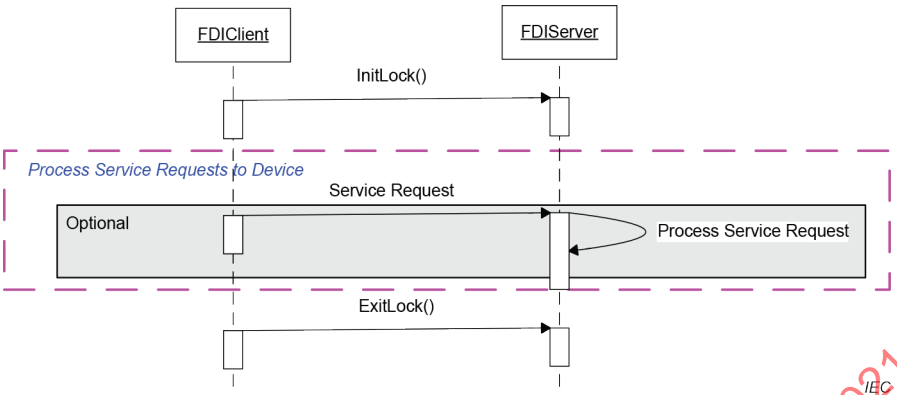

Figure 2 – Locking services

A service request that requires locking shall fail either partially or completely if no lock has been acquired by the FDI Client via InitLock prior to requesting the service. The FDI Client has to release the lock via ExitLock after all service requests have been completed.

A write operation will partially fail, i.e., it will return a status code for each variable in the set of variables to be written since some may belong to devices that are locked and some to devices that are not locked.

FDI Servers may queue InitLock requests until a service for which a lock has been created completes and the lock has been released. However, such an optimization is not part of the standard behavior required of an FDI Server.

5.6 EditContext

5.6.1 Concept and usage model

The FDI Server provides the EditContext model to interact with Clients during their editing task. The concept is closely related to UIDs and fulfills the needs for Server-driven UI dialogs based on EDDL rules.

An EditContext can be used to make changes to Variable Values visible to the Server without applying them to the online or offline representation of a Device. The Server will apply business logic associated to the edited Variable, which – in some cases – causes changes to other Variable Values (e.g. if an engineering unit is changed) or the UID (e.g. a Variable becomes invisible). Thus, the Client can use an EditContext to modify (edit) Parameters such as engineering units, ranges and more, verify any side effects, and re-adjust the settings before applying the changes.

An FDI Server may implement different EditContext strategies:

- A single EditContext instance for all dialogues of an FDI Client.
- Multiple EditContext instances.
- Hierarchical EditContext instances.

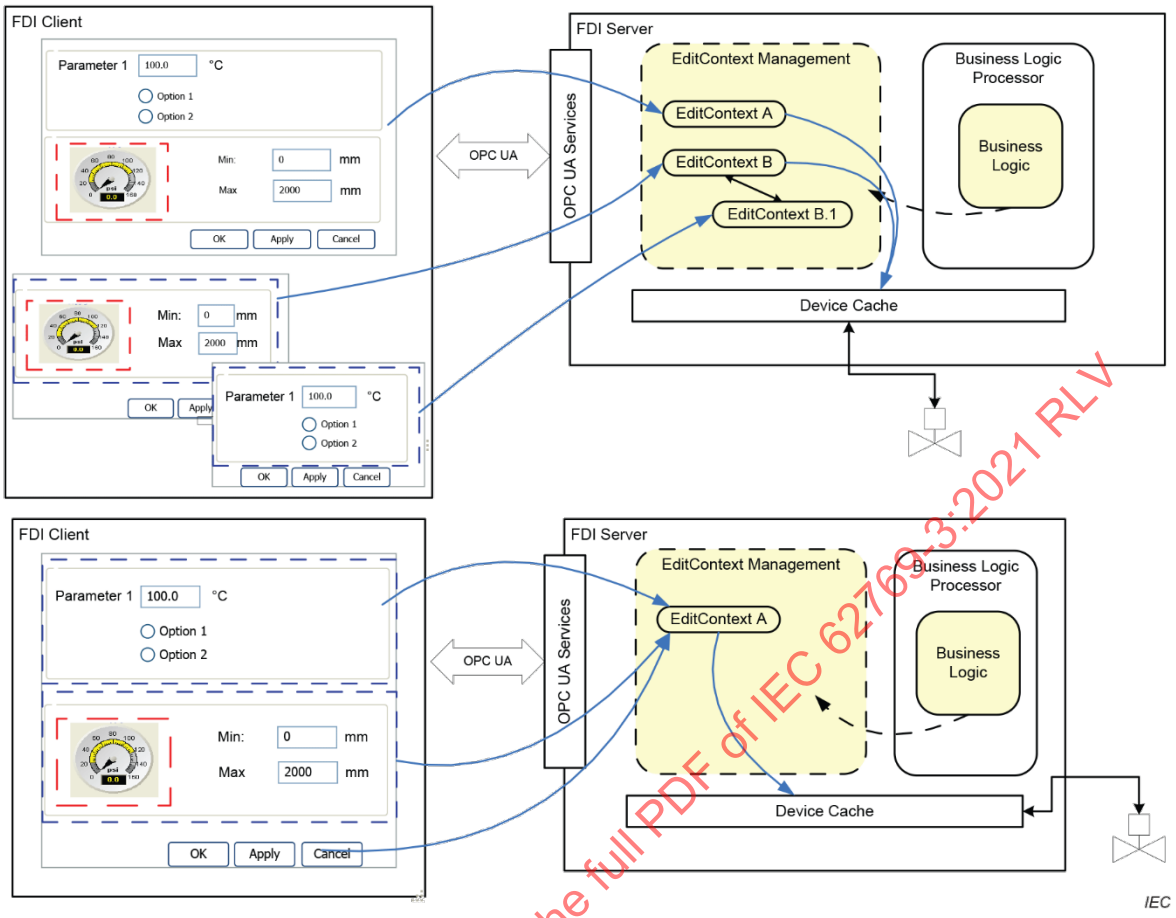

Figure 3 – EditContext models

Figure 3 shows two possible Server strategies and how the Client can adapt. In the lower scenario, the Server provides a single EditContext instance for all dialogs. Here, the Client groups all dialogs and exposes a single set of buttons to Apply and Cancel, because it always concerns all edits.

In the upper scenario, the Server provides multiple EditContext instances, one of them as child of another one. Each instance can be addressed separately. If the changes in a child instance are applied, they are transferred to the parent. If the changes in a root instance are applied, they are transferred to the Device.

The parent-child dependencies are defined in IEC 61804-4:2020, Clause 8.

5.6.2 Services

A set of Services is provided to the FDI Client to maintain EditContext instances (see IEC 62769-5 for a detailed description of these Services):

- GetContext – This Service is used to request an EditContext instance. The Client specifies certain characteristics for the Server to decide which EditContext instance to return. Depending on its internal strategy, the Server returns the same instance or new instances.
- RegisterNodes – The FDI Client has to register all Nodes of the Information Model that shall be maintained in an EditContext. It is possible to register Nodes of the online and of the offline representation of a Device. The result is new NodeIds that the Client shall use when calling Services to read, write, subscribe to Variables or to invoke Actions.

- Apply – Transfer the modified (edited) Variable Values to the parent (either a parent EditContext instance or the Device). If the same Variable has been edited in the parent instance it will be overwritten with a call of the Apply Service for the child.
- Reset – Clears all modifications. A Reset of already applied modifications is not possible.
- Discard – Deletes an EditContext instance (and its children). Edited Values that have not been applied are discarded. Once deleted, all registered NodeIds will be invalid. If such NodeIds are still subscribed, the Client is notified with proper StatusCodes.

The Client first calls GetEditContext to acquire an EditContext instance. It will then register the Nodes it wants to be part of it. The registration returns new NodeIds which can then be used for reading, writing or subscribing Variables and for calling Methods.

The Client can call GetEditContext multiple times, for instance when it opens an additional edit window or for a completely separate dialog (diagnosis in parallel to configuration). It is up to the Server strategy whether it returns a new instance or the same instance. The Client is expected to adapt its user interface to the EditContext strategy of the Server. See Figure 3 for how Clients may position the Apply and Cancel buttons so that the User clearly understands which changes are applied or discarded.

5.6.3 NodeIds

RegisterNode returns two NodeIds for each registered Node: a ContextNodeId and a DeviceNodeId. The Client uses these NodeIds when calling OPC UA Services to read, write and subscribe or call a Method.

Using the ContextNodeId addresses the Value in the EditContext instance. Using the DeviceNodeId addresses the Value in the Device.

5.6.4 Reading

Reading or subscribing a Variable with the ContextNodeId will return the edited Value from the EditContext instance. If no edited Value exists, the Value from the parent instance or the Device (online or offline) will be returned.

The StatusCode indicates whether the Value originates from the Device (StatusCode Good defined in IEC 62541) or from an EditContext instance (StatusCode Good_Edited defined in IEC 62769-5).

Reading or subscribing a Variable with the DeviceNodeId will return the Value from the Device (online or offline).

5.6.5 Writing

Writing to a Variable with the ContextNodeId modifies the Value in the EditContext instance.

Writing to a Variable with the DeviceNodeId modifies the Value in the Device (online or offline). Any edited Values for this Variable in the addressed EditContext instance or its parents will be reset.

5.6.6 Writing dominant and dependent Variables

In some cases, the value of a Variable depends on the value of another Variable. How these dependencies are evaluated is specified in IEC 61804-4.

When such Variables are edited, the FDI Server shall follow the state diagrams specified in Figure 4 for "Online" and Figure 5 for "Offline". These diagrams specify the states and transitions during the editing process of this kind of Variables. Status is the StatusCode that FDI Clients will receive with the Value when monitoring or reading these Variables with the ContextNodeId. For dependent Variables, any Good or Uncertain StatusCode transfers to an Uncertain_DominantValueChanged and a Bad StatusCode to Bad_DominantValueChanged. For dominant Variables, a Good StatusCode transfers into Good_DependentValueChanged, an Uncertain StatusCode into Uncertain_DependentValueChanged and a Bad StatusCode to Bad_DependentValueChanged.

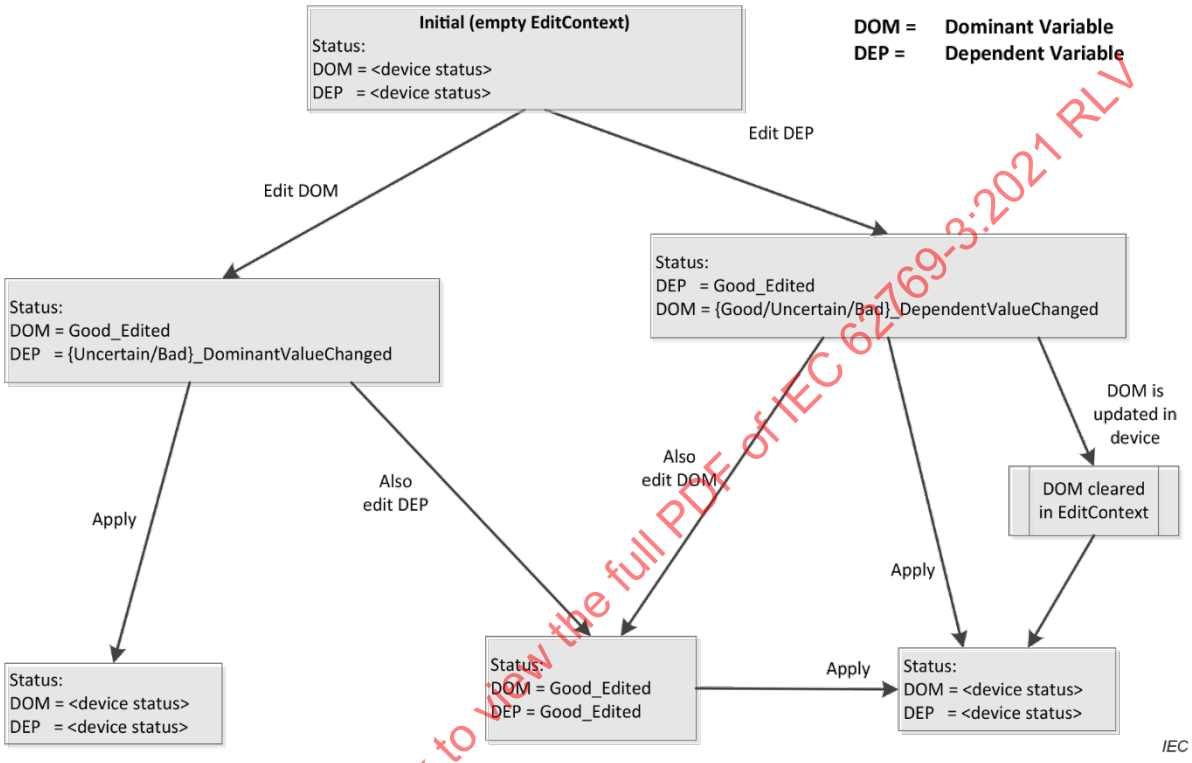

Figure 4 – Online EditContext state diagram for dominant and dependent Variables

If both the dominant and the dependent Variable are to be changed, it is highly recommended for the online case to make these changes in subsequent editing sessions. Systems may enforce this.

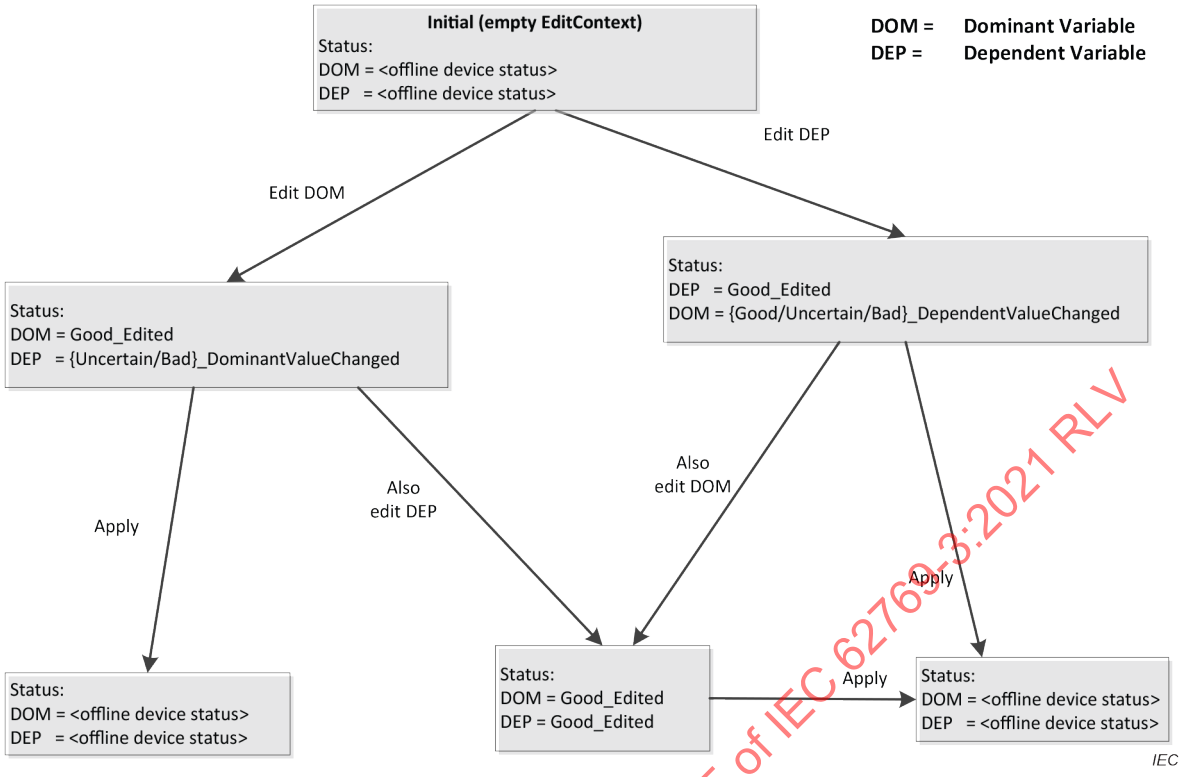

Figure 5 – Offline EditContext state diagram for dominant and dependent Variables

5.6.7 Actions (EDD METHODS)

Before invoking Actions, the Client has to register the ActionSet Node of the Device. The NodeId of this Node has to be specified when calling InvokeAction.

Calling InvokeAction with the ContextNodeId of the ActionSet Node associates it with the proper EditContext instance. The Server will implicitly create an EditContext instance for the invoked Action. This is illustrated in Figure 6.

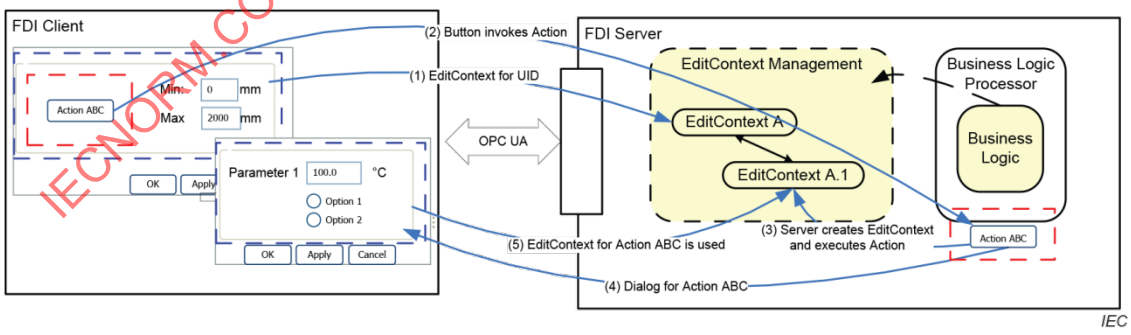

Figure 6 – EditContext for EDD Methods

The EDD METHOD represented by the Action uses builtins to modify Values in the EditContext or the Device, to synchronize changes with the underlying cache or to discard them.

If the Action execution fails, the EditContext for the Action is discarded.

5.6.8 UIDs

The UID Interpreter in the Client calls GetEditContext before it calls up a top-level UID. Additional EditContexts for dialogs may be instantiated by the Server and passed to the Client inside each UID document. See IEC 62769-2 for the UID Schema and the handling of an EditContext in the UID Interpreter.

5.6.9 Synchronization

A Lock has to be created before the first Value is written to an EditContext.

Locking is also required when writing directly to the Device.

5.7 Reading

5.7.1 General

The Read service specified in IEC 62541-4 can be used to read a single value or multiple values from a single device or multiple devices. If a Read service request specifies multiple values are to be read, the values are read in the order they appear in the service request.

All values that are returned to the FDI Client as result of a Read service request shall be unscaled.

A failure encountered while reading a single value shall not abort the read process; all values shall be read. Each value returned contains a status indicating success or failure. Standard OPC UA service status information is returned by the FDI Server as a result of the service calls as specified in 6.2.

An FDI Package can define read actions that are executed by the FDI Server during read service requests. These actions shall not require user interaction; they are strictly intended to be used for Business Logic processing. Any read action that eventually requires user interaction will not perform the builtin but will return an error if possible. The following read actions may be defined in an FDI Package:

- Pre-read Actions
- Post-read Actions

The FDI Server invokes pre-read and post-read actions during the processing of read service requests of online values only; they are not invoked when reading offline values.

In addition to read actions, an FDI Package can define refresh actions that are executed by the FDI Server during read service requests. These actions shall not require user interaction; they are strictly intended to be used for Business Logic processing. Any refresh action that eventually requires user interaction will not perform the builtin but will return an error if possible.

The FDI Server invokes refresh actions during the processing of read service requests of both offline and online values.

The refresh actions mentioned in 5.7.1 are not to be mixed up with refresh relations.

NOTE Refresh actions are defined by means of the EDDL REFRESH_ACTIONS construct inside an EDDL VARIABLE construct. On the other hand, refresh relations are defined by means of the EDDL REFRESH construct. The handling of refresh relations is included in the generic event "Process Conditionals/Relations" that appear in the sequence diagrams for read, write and subscription services. The explanations that follow the diagrams include refresh relations in the general term "EDDL relations". See IEC 61804-3 for more details on both refresh actions and refresh relations.

5.7.2 Reading offline variables

The sequence diagram in Figure 7 shows the behavior of the FDI Server when an offline value is read.

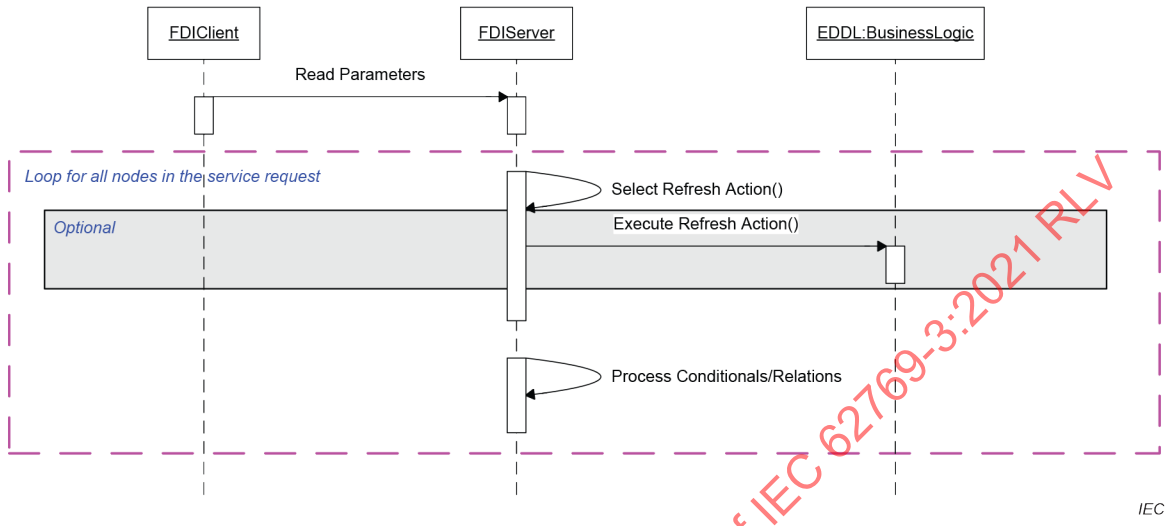

Figure 7 – Offline variable read

If a variable has refresh actions associated with it, the FDI Server always executes those actions regardless of MaxAge.

If the refresh actions fail, the status returned for that variable shall indicate the read failed.

The FDI Server evaluates conditionals and relations after the refresh actions are executed. This provides an opportunity for the re-evaluation of conditional expressions in EDDL Business Logic and the processing of EDDL relations.

5.7.3 Reading online variables

The FDI Server can cache the online values read from a device. The FDI Server maintains a timestamp for each online value that indicates when the value was read from the device. The FDI Server uses the MaxAge argument of a Read service request to determine whether the cached value can be returned. If the difference between the timestamp and the current time exceeds the MaxAge argument, the FDI Server shall read the value from the device. Otherwise, the cached value can be returned.

Read actions are only executed when the variable is read from the device.

The sequence diagram in Figure 8 shows the behavior of the FDI Server when an online value is read.

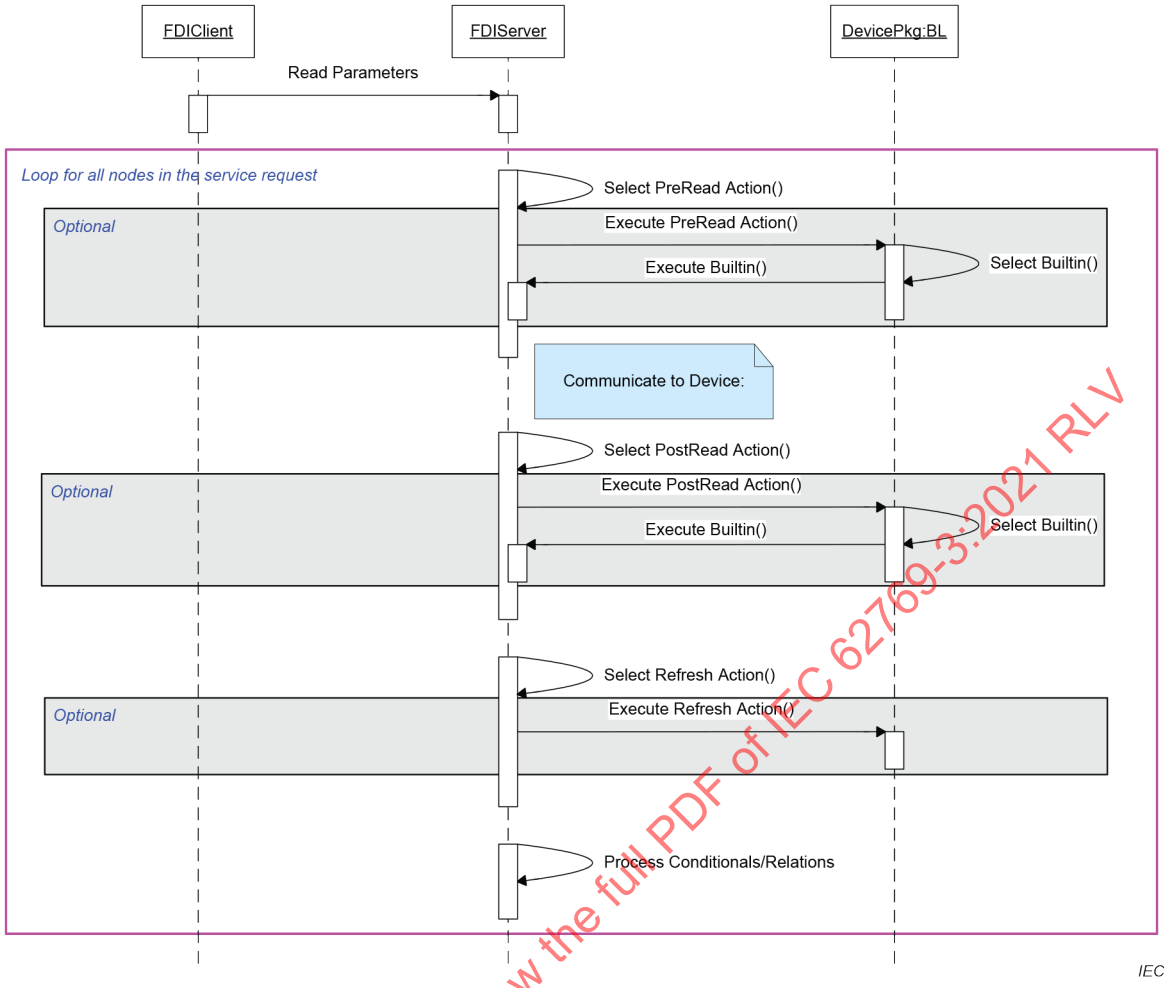

Figure 8 – Online variable read

If a variable has pre-read actions associated with it, these actions are executed prior to reading the variable from the device. If a variable has post-read actions associated with it, these actions are executed after reading the variable from the device. If the pre-read or post-read actions fail, the status returned for that variable shall indicate the read failed.

If a variable has refresh actions associated with it, these actions are handled as in the offline variable read case (see 5.7.2).

The FDI Server evaluates conditionals and relations after post-read actions are executed. This provides an opportunity for the re-evaluation of conditional expressions in EDDL Business Logic and the processing of EDDL relations.

5.8 Writing

5.8.1 General

The Write service specified in IEC 62541-4 can be used to write a single value or multiple values to a single device or multiple devices. If a Write service request specifies multiple values are to be written, the values are written in the order they appear in the service request.

A failure encountered while writing a single value shall not abort the write process; all values shall be written. A status is returned indicating success or failure of each value included in the service request. Standard OPC UA service status information is returned by the FDI Server as a result of the service calls as specified in 6.2.

Unlike the read operation, write failures when multiple variables are specified may leave the device in an indeterminate state with some variables modified and others left unmodified. It is up to the FDI Client to handle partial failures.

FDI Clients need to lock the device for exclusive access prior to writing. The lock request may be issued immediately before the write service request or it may be issued independently across multiple write service requests (see 5.5).

The FDI Server performs data validation during write service requests of online and offline values.

An FDI Package can define write actions that are executed by the FDI Server during write service requests. These actions shall not require user interaction; they are strictly intended to be used for Business Logic processing. Any write action that eventually requires user interaction will not perform the builtin but will return an error if possible. The following write actions may be defined in an FDI Package:

- Pre-write Actions
- Post-write Actions

The FDI Server invokes those actions during the processing of write service requests of online values only; they are not invoked when writing offline values.

5.8.2 Write offline variables

The sequence diagram in Figure 9 shows the behavior of the FDI Server when an offline value is written.

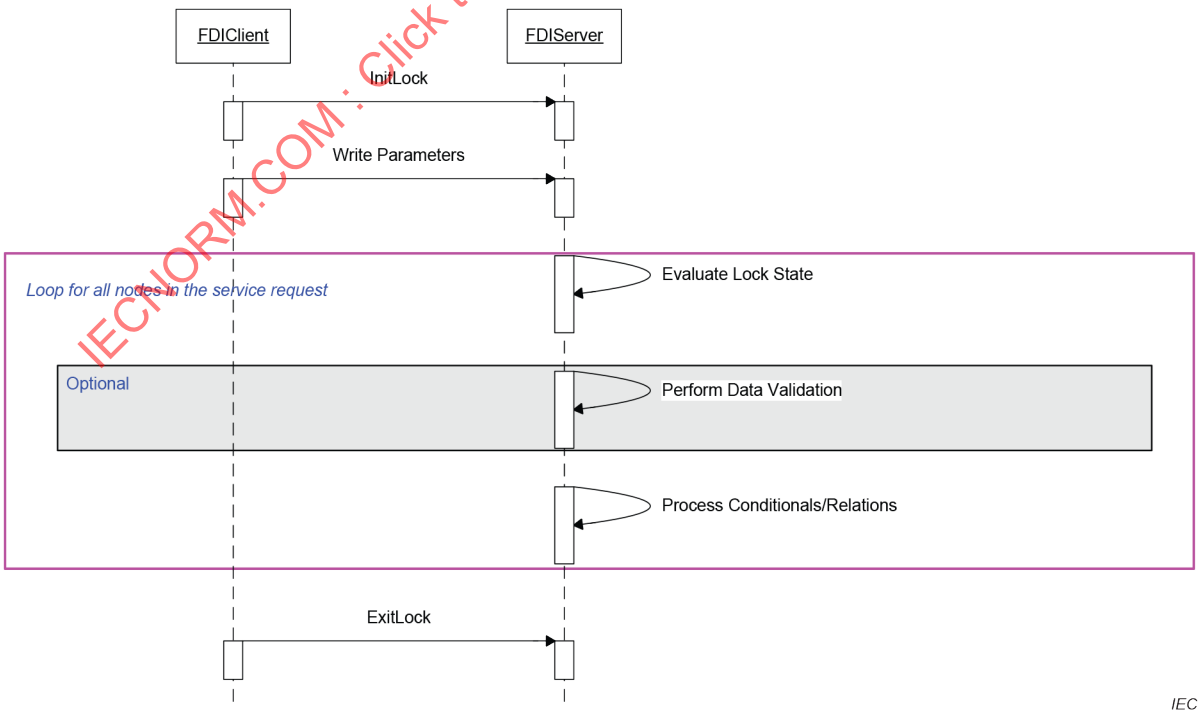

Figure 9 – Offline variable write immediate

As a starting point for writing a variable, the FDI Server verifies if the device is locked by the FDI Client. If it is not locked, the status returned for that variable shall indicate the write failed.

If the device is locked by the FDI Client, the FDI Server performs data validation. The validation consists basically of range and type check based on EDDL information. If the type validation fails, the status returned for that variable shall indicate the write failed. If the range validation fails, the status returned for that variable shall indicate the write succeeded, but the status information of the variable value in the Information Model shall indicate that it is bad, out-of-range.

If the validation process succeeds, the FDI Server writes to the offline value of the variable in the Information Model.

After writing the variable value, the FDI Server evaluates conditionals and relations. This provides an opportunity for the re-evaluation of conditional expressions in EDDL Business Logic and the processing of EDDL relations.

5.8.3 Writing online variables

The sequence diagram in Figure 10 shows the behavior of the FDI Server when an online value is written.

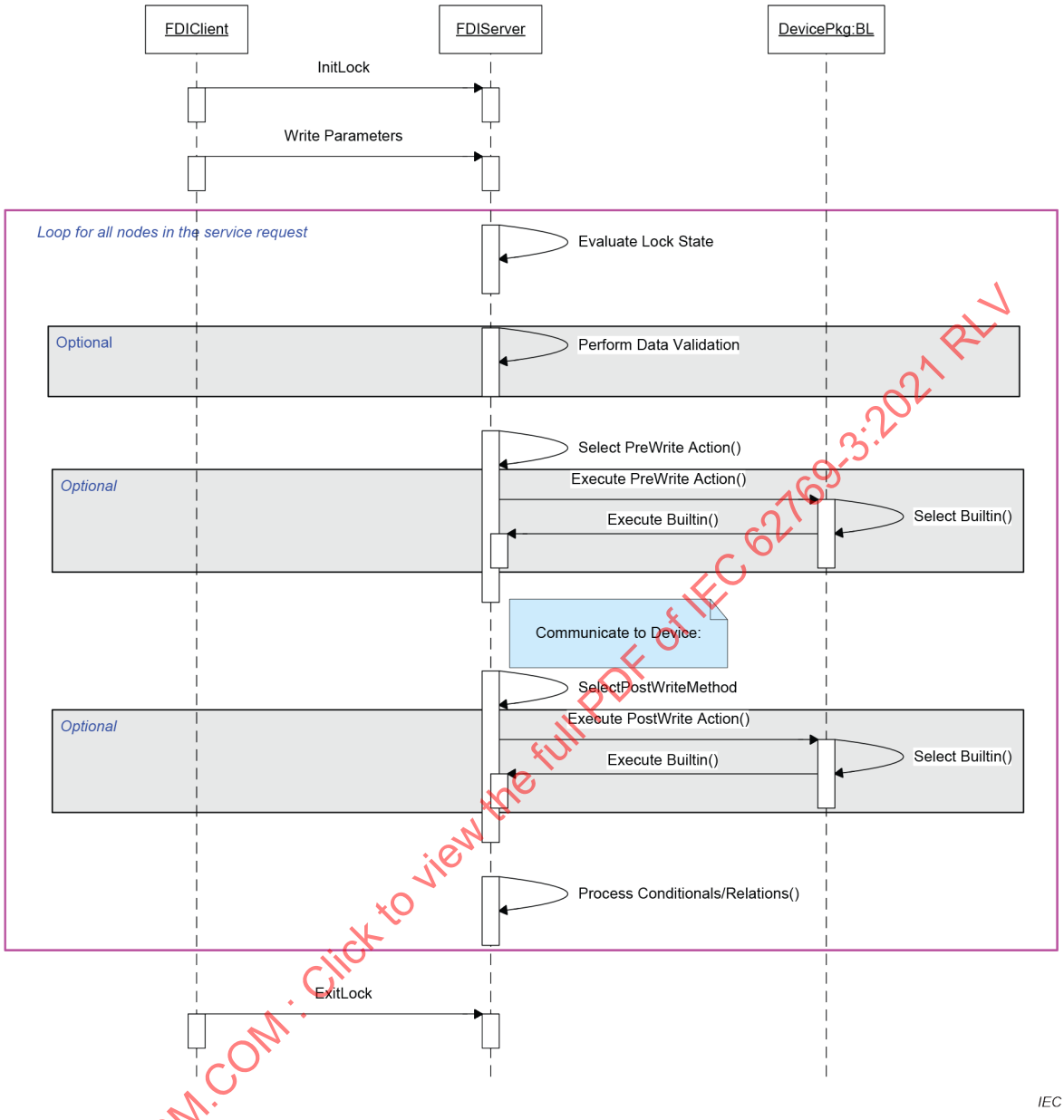

Figure 10 – Online variable write immediate

When writing an online variable, the FDI Server verifies if the device is locked by the FDI Client and performs data validation as described in 5.8.2.

If the validation process succeeds, the FDI Server writes the variable to the physical device.

If a variable has pre-write actions associated with it, these actions are executed prior to writing the variable to the device. If the pre-write actions fail, the status returned for the variable shall indicate the write failed and write operation terminates without writing to the device.

If a variable has post-write actions associated with it, these actions are executed after writing the variable to the device. If the post-write actions fail, the status returned for the variable shall not indicate the write failed, since the value has already been written to the device. The status returned shall be Good_PostActionFailed.

The FDI Server evaluates conditionals and relations after post-write actions are executed. This provides an opportunity for the evaluation of conditional expressions in EDDL Business Logic and the processing of EDDL relations.

5.8.4 Writing to an EditContext

The EditContext is specified in 5.6.

The sequence diagram in Figure 11 shows the general behavior of an EditContext when Values are edited and applied.

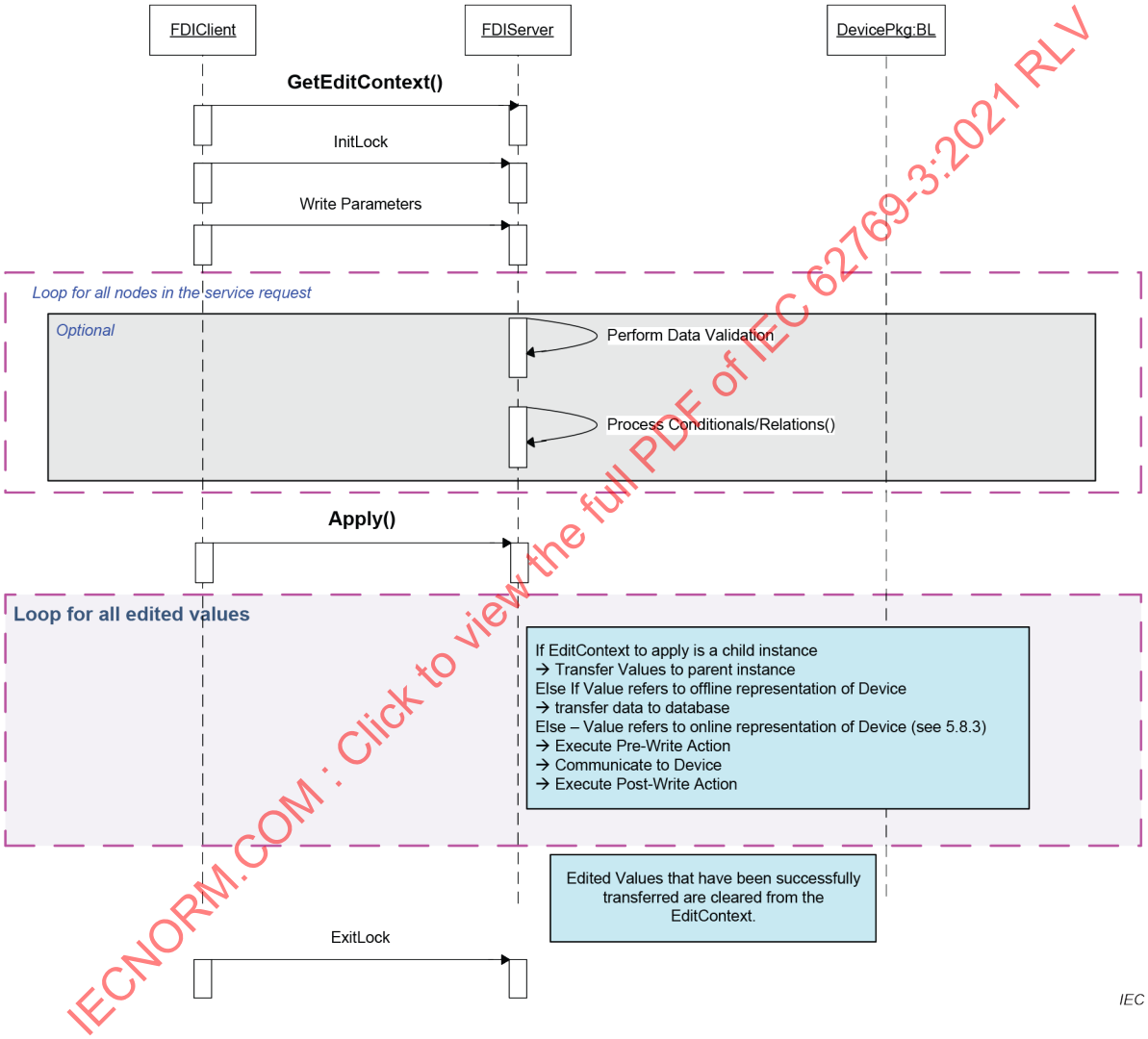

Figure 11 – Write with EditContext

When writing Variables to an EditContext, the FDI Server performs data validation as in the normal writes to online or offline data. It also processes Conditionals/Relations. Changes to other Variables resulting from this process are also written to the EditContext.

When calling Apply, the modified Variables are transferred to the parent. If the parent is the Device and a variable has pre-write actions associated with it, these actions are executed prior to writing the variable to the Device. If the pre-write actions fail, the status returned for Apply shall indicate the error.

If the parent is the Device and a variable has post-write actions associated with it, these actions are executed after writing the variable to the Device. If the post-write actions fail, the status returned for the variable shall not indicate the write failed, since the value has already been written to the device. The status returned shall be Good_PostActionFailed.

5.9 Subscription

5.9.1 General

The Subscription service specified in IEC 62541-4 can be used to initiate the monitoring of a single variable or multiple variables from a single device or multiple devices.

A failure related to a single variable while establishing a subscription to multiple variables shall not abort the subscription process; all variables shall be monitored. Each variable returned contains a status indicating success or failure. Standard OPC UA service status information is returned by the FDI Server as a result of the service calls, as specified in 6.2.

An FDI Package can define read actions that are executed by the FDI Server during the monitoring process of a subscription. These actions shall not require user interaction; they are strictly intended to be used for Business Logic processing. Any read action that eventually requires user interaction will not perform the builtin but will return an error if possible. The following read actions may be defined in an FDI Package:

- Pre-read Actions
- Post-read Actions

The FDI Server invokes these actions during the monitoring of online variables only; they are not invoked when monitoring offline variables.

In addition to read actions, an FDI Package can define refresh actions that are executed by the FDI Server during the monitoring process. The FDI Server invokes these refresh actions during the monitoring of both offline and online variables. These actions shall not require user interaction; they are strictly intended to be used for Business Logic processing. Any refresh action that eventually requires user interaction will not perform the builtin but will return an error if possible.

The sampling interval requested by the FDI Client and established by the FDI Server defines a time interval that is used to periodically check for changes in the variables' value or status. At each time interval, the actions are invoked, and the value and status are compared with the previous value and status. A change in the value or status will result in the FDI Server preparing a notification of the new value and status.

5.9.2 Subscription of offline variables

The sequence diagram in Figure 12 shows the behavior of the FDI Server when an offline value is being monitored.

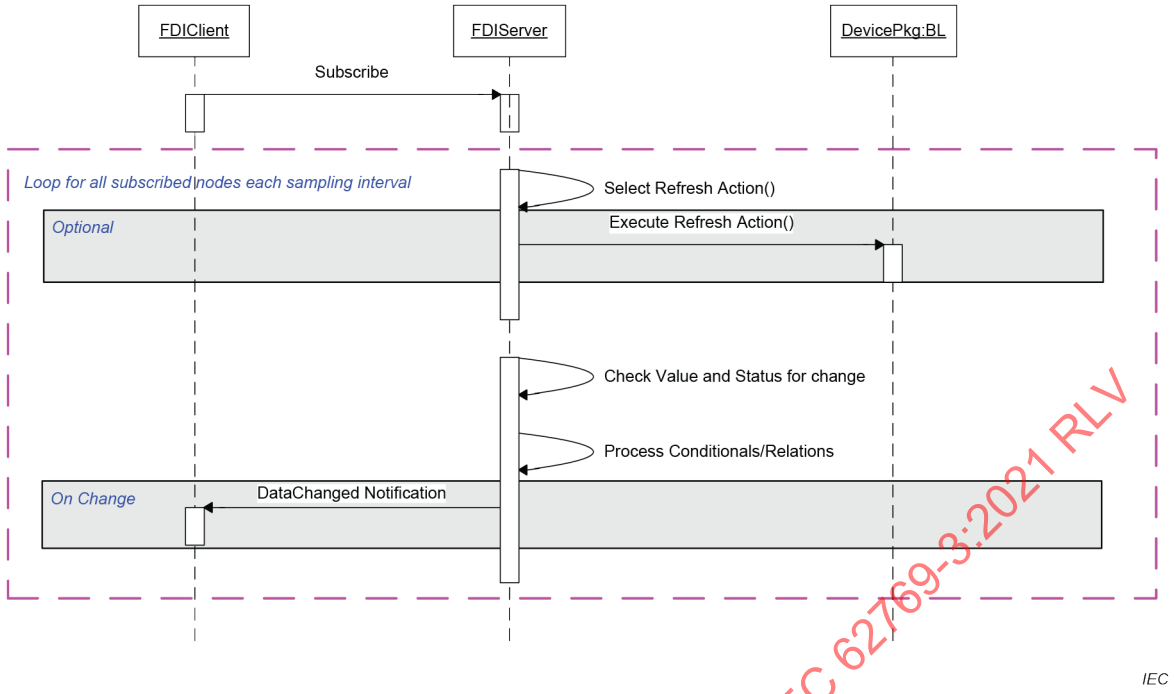

Figure 12 – Offline variable subscription

If a variable has refresh actions associated with it, the FDI Server executes those actions at each time interval.

The FDI Server evaluates conditionals and relations after the refresh actions are executed.

5.9.3 Subscription of online variables

The sequence diagram in Figure 13 shows the behavior of the FDI Server when an online variable is being monitored.

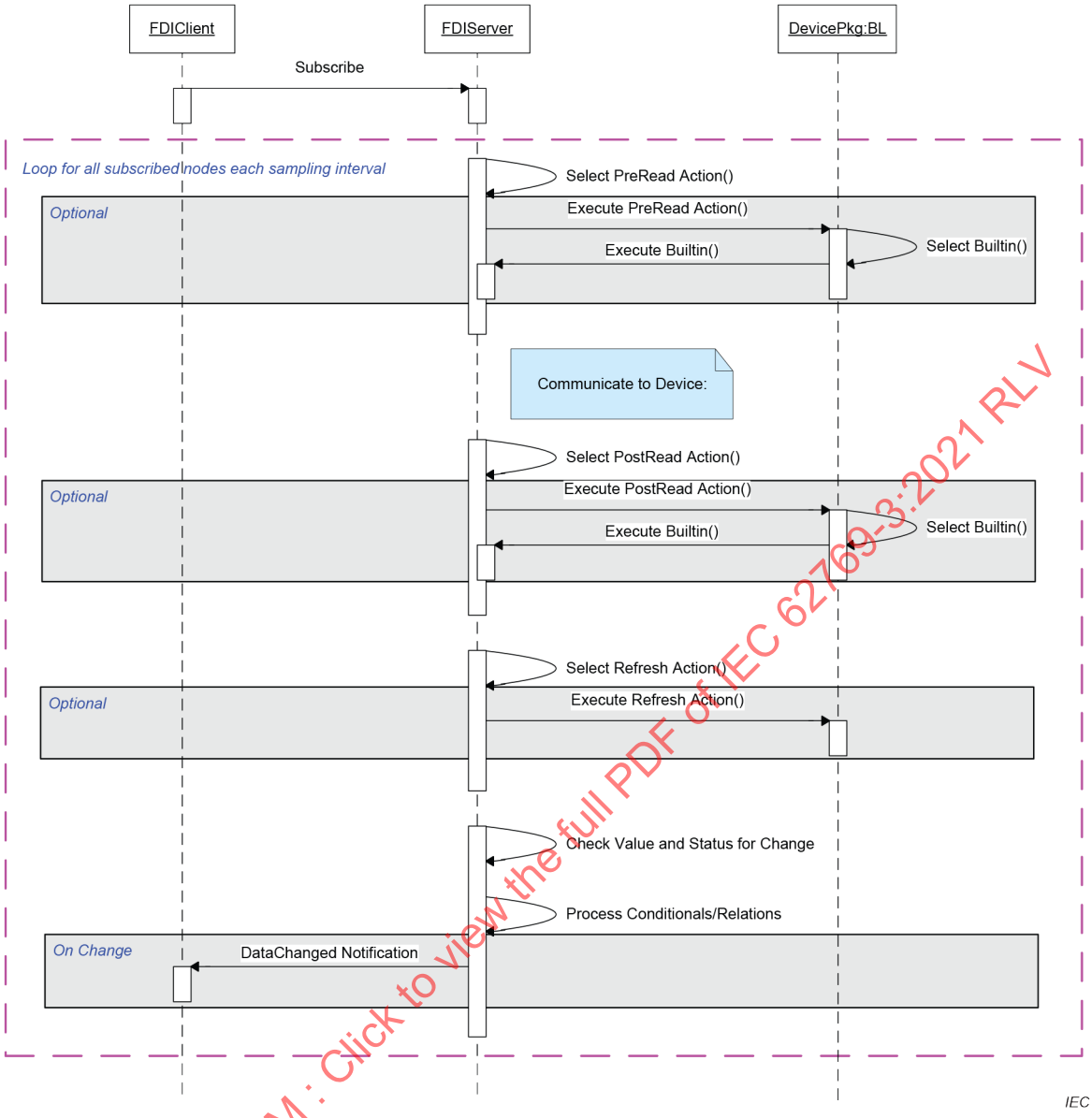

Figure 13 – Online variable subscription

The variable is read from the device at each time interval.

If a variable has pre-read actions associated with it, these actions are executed prior to reading the variable from the device.

If a variable has post-read actions associated with it, these actions are executed after reading the variable from the device.

If a variable has refresh actions associated with it, the FDI Server executes those actions after the post-read actions.

The FDI Server evaluates conditionals and relations after any associated actions are executed.

If the actions fail or the device read fails, the status of the variable shall indicate the failure.

5.10 Device topology

5.10.1 General

The FDI Server maintains Device Instances in the Information Model. The Information Model maintained by the FDI Server reflects the structure of the system; the FDI Server maintains device information in the context of the Device Topology.

The Device Topology includes devices, connecting communication networks, and the elements to communicate via these networks. The Device Topology is defined in IEC 62769-5; Objects, References and the AddressSpace organization required to create the proper Information Model are defined as part of the Information Model specification.

5.10.2 Connection Points

The following non-normative Figure 14 illustrates the topology within the Information Model.

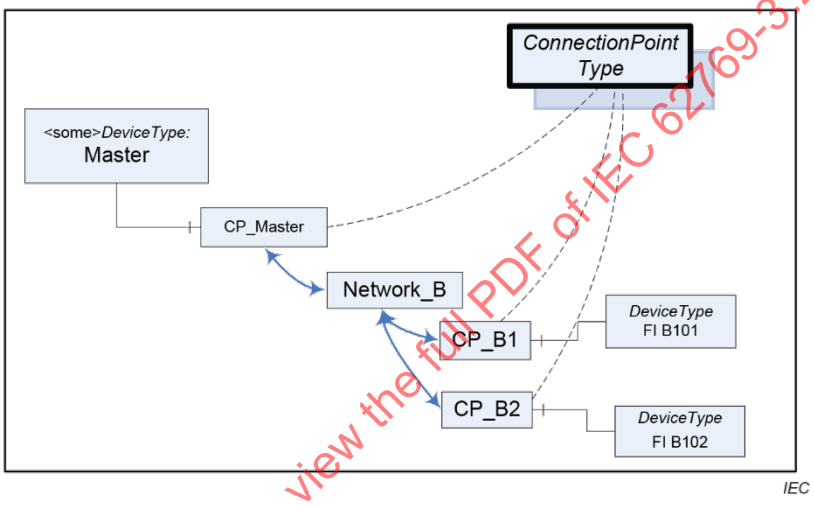

Figure 14 – Topology with Network objects (non-normative)

The FDI Server uses the information in the FDI Package to create both the type definitions for the devices and Communication Devices as well as their respective Connection Point elements. The mapping between the FDI Package definition and the Information Model elements is defined in IEC 62769-5.

Network types for the protocols that are supported by Native Communication devices are provided by the FDI Server. Network types for non-native protocols are provided through FDI Package definitions provided for the communication server.

Device definitions contain one or more Connection Point definitions for a device. Each Connection Point maintains a reference to a protocol element that specifies the protocol for the Connection Point. A device may be capable of providing or using multiple protocols; each protocol provided or supported will have a unique Connection Point. The network objects contain a reference to a protocol definition element that defines the protocol utilized by the network object.

The Device and Connection Point type definitions are used to create Device and Connection Point instances. The network type definitions are used to create instances of networks in the system. The network types are specified in the protocol-specific annexes in IEC 62769-4, and are provided by the FDI Server as an integral component. The FDI Server, or FDI Clients interacting with the FDI Server to create the topology, locates devices, communication devices, connection points, and networks using standard OPC UA entry points for browsing:

- DeviceSet – references Device Instances in the FDI Server;
- CommunicationSet – references instances of communication devices in the FDI Server;
- NetworkSet – references instances of networks in the FDI Server.

The FDI Server, or FDI Clients interacting with the FDI Server to create the topology, creates references between instances of a device, an associated Connection Point, and a network element (see Figure 14). The FDI Server validates that the protocol associated with the Connection Point and the protocol associated with the network are of the same protocol type. If a device defines multiple Connection Points, the FDI Server will use the Connection Point of the device that matches the network protocol.

Each network object shall be associated with at least one Communication Device. The association between the communication device and the network element shall be done before devices can be associated with the network element. Once a Communication Device is associated with a network element, Business Logic in the Communication Device will be used for network management.

The network element definition specifies the number of Connection Points that can be added to the network.

The references established between devices, Connection Points, and networks do not affect the reference established for the standard browse entry points. Devices remain referenced by DeviceSet regardless of whether the device has a reference to a Connection Point or to a network.

5.10.3 Topology management

5.10.3.1 General

FDI Server vendors have two options to provide trusted FDI Clients with the ability to manage the topology:

a) they may provide vendor-specific functionality;

b) they may implement the OPC UA NodeManagement Service Set.

If the FDI Server vendor chooses the second option, i.e. implementing the OPC UA NodeManagement Service Set, the topology management shall be implemented as specified in 5.10.3.

In order to prevent simultaneous access from different agents that are trying to modify the topology, the elements involved in the topology modification are locked. The scope of the lock for Modular Devices and networks is specified in IEC 62769-5.

The FDI Package for the Communication Device includes definitions that are used by the FDI Server to manage and validate the topology, including the optional action ValidateNetwork (see IEC 62769-7). Those definitions are used by the FDI Server during topology management.

5.10.3.2 Add Device to Network

The sequence diagram in Figure 15 shows the behavior of the FDI Server when a device is added to a network.

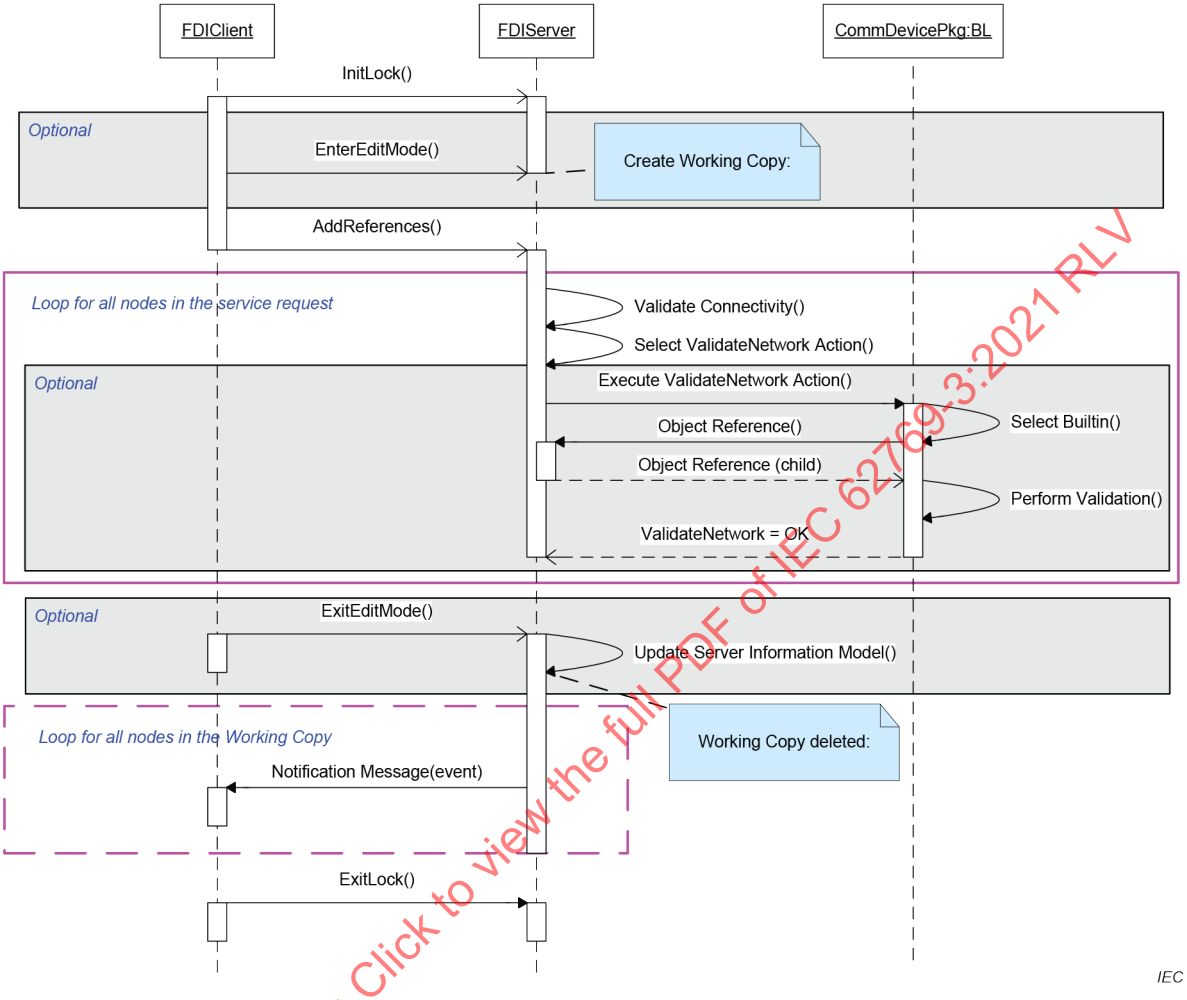

Figure 15 – Add Device to topology

The AddReferences service specified in IEC 62541-4 is used to add devices to the topology. The AddReferences service can be used to establish a single or multiple references. If an AddReferences service request specifies multiple references are to be added, the references are added in the order they appear in the service request.

A failure encountered while adding a single reference shall not abort the entire process; all references shall be processed. A status is returned indicating success or failure for each reference included in the service request. Standard OPC UA service status information is returned by the FDI Server as a result of the service calls, as specified in 6.2.

When adding a reference, the association of a device's Connection Point with a network shall be validated by the FDI Server; this applies to both FDI Server vendor-specific topology management as well as the OPC UA AddNode method.

The FDI Server performs an initial validation of the connectivity, which includes, for instance, verifying that the protocol specified by the Connection Point and the network are the same and the number of connections supported by the network have not been exceeded. That validation is based on information provided with the FDI Communication Package of the involved elements. If the initial validation succeeds, the ValidateNetwork Action provided by the Communication Device is invoked by the FDI Server. See IEC 62769-7 for more details.

In a first pass, all requested references are added regardless of the validation process succeeding or failing. After adding all references, a second validation pass is done. If the validation process fails for any reference, that reference shall not be added, and the status returned for that reference shall indicate the reference failed to be added.

Because of this two-step validation process performed by the FDI Server, the result of adding multiple references at a time can be different from the result of adding one reference at a time.

5.10.3.3 Remove Device from Network

The sequence diagram in Figure 16 shows the behavior of the FDI Server when a device is removed from a network.

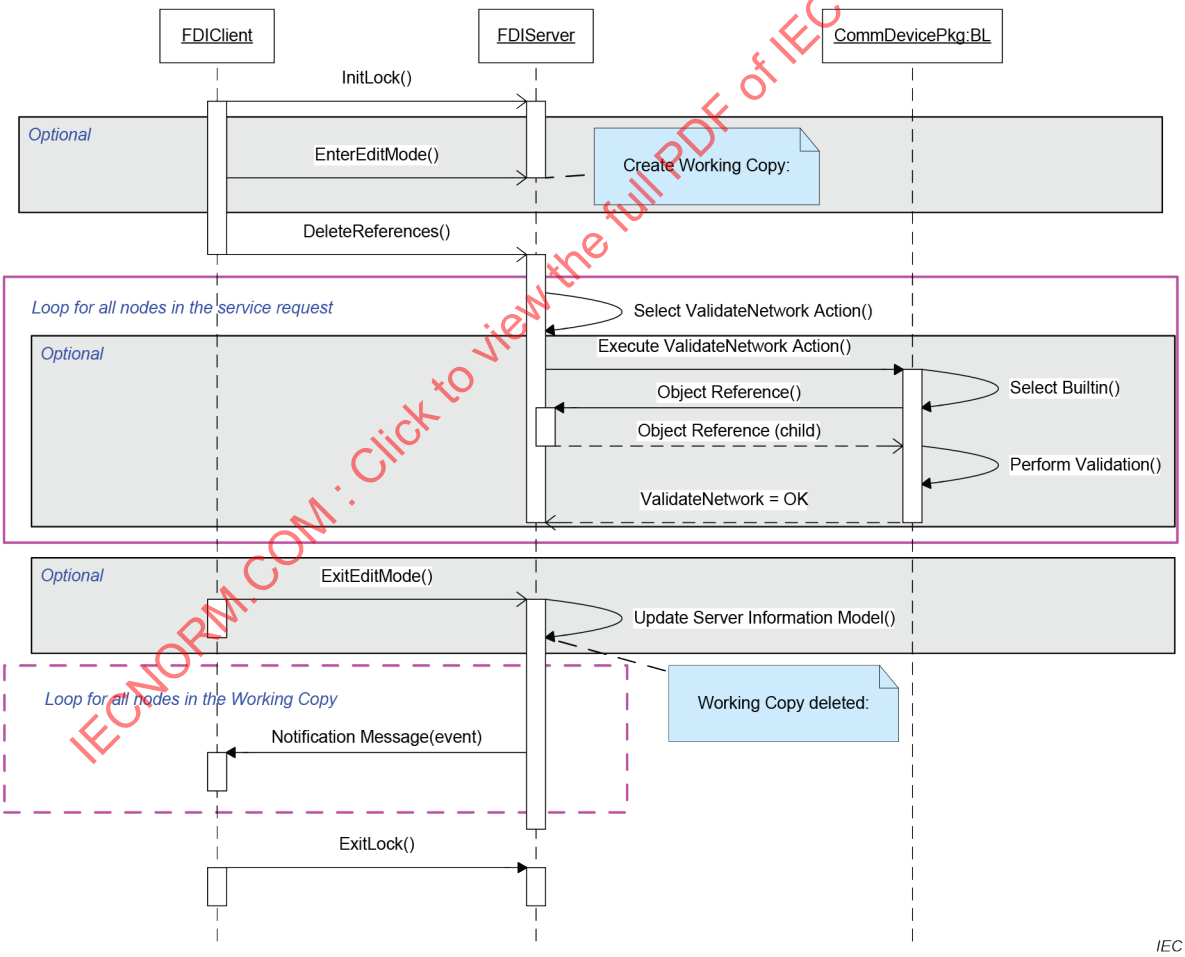

Figure 16 – Remove device from topology

The DeleteReferences service specified in IEC 62541-4 is used to remove devices from the topology. The DeleteReferences service can be used to remove a single reference or multiple references. If a DeleteReferences service request specifies multiple references are to be removed, the references are removed in the order they appear in the service request.

A failure encountered while removing a single reference shall not abort the entire process; all references shall be processed. A status is returned indicating success or failure for each reference included in the service request. Standard OPC UA service status information is returned by the FDI Server as a result of the service calls, as specified in 6.2.

The removal of a device from a network requires the FDI Server to validate the network. The FDI Server invokes the ValidateNetwork Action provided by the Communication Device after removal of the device from the network.

In a first pass, all requested references are removed regardless of the validation process succeeding or failing. After removing all references, a second validation pass is done. If the validation process fails for any reference, that reference shall not be removed, and the status returned for that reference shall indicate the reference failed to be removed.

Because of this two-step validation process performed by the FDI Server, the result of adding multiple references at a time can be different from the result of adding one reference at a time.

5.10.4 Topology scanning

The sequence diagram in Figure 17 shows the behavior of the FDI Server when the topology is scanned.

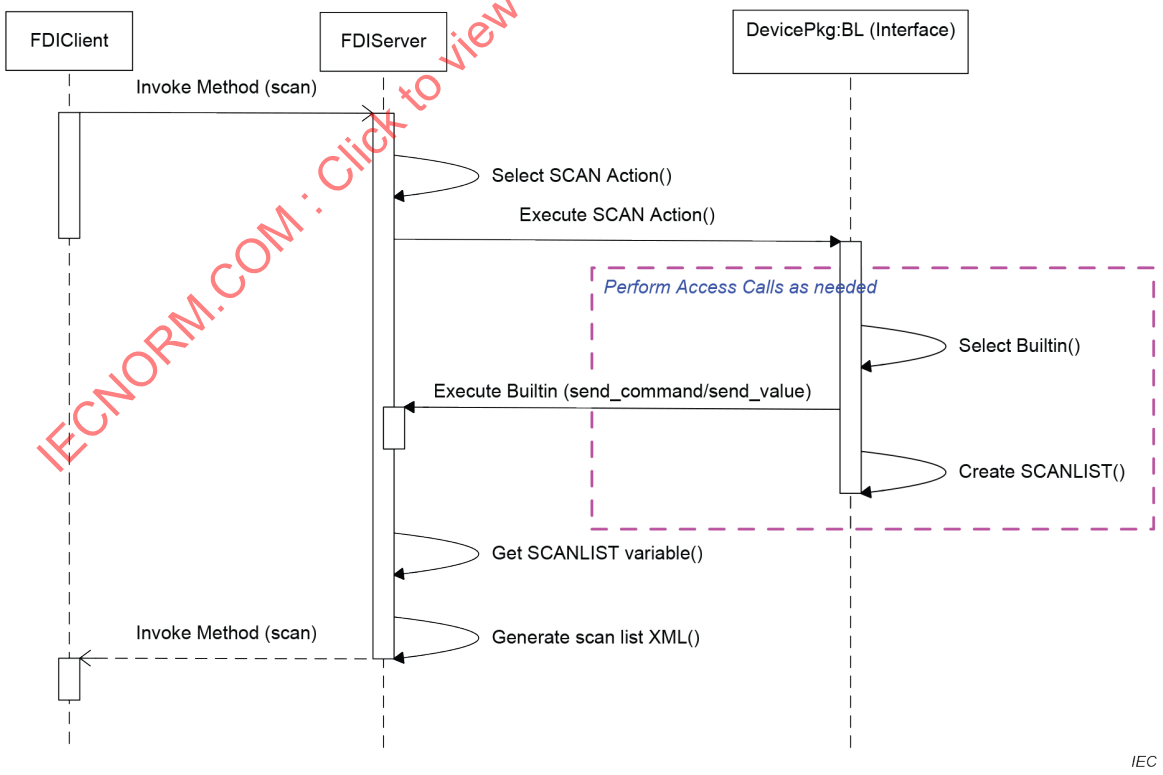

Figure 17 – Scan topology

Scanning of a network for connected devices is provided via the Scan Service associated with Communication Devices. This service is described in IEC 62769-5.

For nodes representing Native Communication devices, the FDI Server provides the service implementation.

For Communication Devices that are not native to the FDI Server, the FDI Server invokes the SCAN Action provided by the Communication Device (see IEC 62769-4). The SCAN action invokes builtins provided by the FDI Server to send commands to the Communication Device. The SCAN action processes the responses to the commands to create a scan list.

The scan list created by the SCAN Action is stored in a DDLIST variable referenced through the SCAN_LIST variable. The DDLIST contains the definition for the devices detected by the communication device.

The Information Model specified in IEC 62769-5 provides protocol independent definitions for devices. The protocol independent device definitions contain references to nodes containing protocol-specific identification for a device.

The DDLIST variable resulting from the scan is composed of variable definitions. The DDLIST will contain variables whose name matches the properties and attributes defined in IEC 62769-5 for the protocol independent device definition. These protocol-independent definitions allow the FDI Server to formulate protocol-independent information to an FDI Client about the devices in the network.

The SCAN Action may not fully identify a device; variables providing protocol-independent information may be missing from the DDLIST. The FDI Server through vendor-specific functions performs additional parameter read and write actions to the device to complete the identification. This functionality is both FDI Server vendor-specific as well as protocol-specific and is not standardized.

The DDLIST also contains network addressing information for the devices identified in the network. The network addressing will be protocol-specific but match the protocol-specific Connection Point properties and attributes specified in IEC 62769-5.

The DDLIST may also contain additional variables that are protocol-specific. The protocol-specific variables are standardized by the foundations defining the network protocol, see protocol-specific annexes in IEC 62769-4.

The FDI Server is responsible for the creation of the XML data set returned by the Scan Service using the information provided by the DDLIST variable.

5.10.5 Use of SCAN function

The SCAN function can be used by the FDI Server as part of topology management. The FDI Server vendor-specific functions for topology management may perform network SCANS to define the Device Instances to create and to initialize the Information Model topology.

The SCAN function is provided by communication devices; a device definition does not have to exist in the Information Model for the SCAN function to succeed. The information provided by the SCAN function may be used by FDI Server vendor-specific functionality to create the Device Topology. FDI Server vendor-specific functionality is responsible for matching the devices identified by the SCAN function to device types in the Information Model.

The FDI Server, through vendor-specific implementation, uses the SCAN function as part of commissioning a network. The FDI Server vendor-specific implementation allows the FDI Server to match and validate the offline created topology against the physical network (see 5.10.6).

The use of the SCAN function for topology creation and topology validation is not standardized and is an FDI Server vendor-specific functionality.

5.10.6 Validation of defined topology

The FDI Server shall validate that the defined Device Instance in the Information Model matches the physical device. The FDI Server vendor-specific functionality is responsible for validating the Information Model against the physical devices connected in the system.

The FDI Server can rely on standard functions such as SCAN to identify the devices physically connected and determine whether there is a match with the offline defined topology. The FDI Server may also use protocol-specific commands to identify devices.

The FDI Server shall validate that the physical device is of the same type as the Device Instance in the Information Model. The FDI Server vendor-specific implementations can allow connectivity to devices of different revisions or require an exact match. FDI Server vendor-specific functionality provides the device matching.

5.11 User Interface Elements

5.11.1 User Interface Descriptions

User Interface Descriptions (UIDs) are descriptive user interfaces that are rendered by an FDI Client. They appear in the Information Model as UIDescriptionType nodes (see IEC 62769-5).

FDI Clients retrieve UIDs by reading the Value attribute of a UIDescriptionType node in the Information Model. The Value attribute of a UIDescriptionType node contains the UID in the form of an XML string (see IEC 62769-2).

Any values that the FDI Server provides to the FDI Client through the UID shall be unscaled, including, for instance, current variable values, ranges and initial values.

FDI Servers shall evaluate any conditional behavior present in a UID before providing it to the FDI Client. The FDI Client shall simply render the UID provided by the FDI Server.

This example assumes the UID is based on EDDL. If the PATH attribute of an IMAGE is conditional (i.e., dependent on the value of a parameter in the device), the FDI Server shall evaluate the conditional to determine which image is to be used and then provide the appropriate Browse Name to the FDI Client via the XML string. The FDI Client will simply render the appropriate image. All other EDDL conditionals shall be evaluated by the FDI Server in a similar fashion.

An FDI Package can define actions that are associated with UIDs. The following actions may be defined in an FDI Package:

- Pre-edit Actions;
- Post-edit Actions;
- Init Actions;
- Refresh Actions;
- Exit Actions.

Those actions are executed by the FDI Server, but their execution is driven by the FDI Client.

In order to allow the FDI Client to drive the execution of actions, the FDI Server creates Actions Proxies (see 5.12.3) and makes the Action Proxies names available to the FDI Client by means of the ListOfActions element type defined in the XML schema (IEC 62769-2). The FDI Server thus maintains the Actions Proxies names in the XML string of the UID.

As the FDI Client retrieves UIDs, it retrieves the Actions Proxies names associated to them as well.

Even though a single EDD Actions definition may specify more than one EDD Method, the FDI Server does not provide individual references to each EDD Method that is specified, but it provides a single Actions Proxy name to refer to all EDD Methods specified in the EDD Actions clause. As a consequence, the list of actions specified in the XML schema will always have a single entry.

As the FDI Client processes a UID, it can start the execution of actions by calling the InvokeAction service and passing the corresponding Actions Proxy name as argument (see 5.12.3).

5.11.2 User Interface Plug-ins

User Interface Plug-ins (UIPs) are programmatic user interfaces that are executed by an FDI Client. They appear in the Information Model as UIPluginType nodes (see IEC 62769-5).

FDI Clients retrieve UIPs by reading the Value attribute of a UIPluginType node in the Information Model. The Value attribute of a UIPluginType node is a byte array containing a binary executable component (see IEC 62769-5).

FDI Packages can provide multiple variants of the same UIP (see IEC 62769-4). FDI Clients browse through the available UIP Variants and select the variant that is most appropriate.

Unlike UIDs, UIPs are not processed by an FDI Server in any way; they are imported from the FDI Package and simply provided to the FDI Client upon request.

FDI Device Packages can reference a UIP in a separate FDI UIP Package (see IEC 62769-4). FDI Servers shall resolve these references. Any references that cannot be resolved shall result in a Bad_NodeIdUnknown status code when the UIP is read by an FDI Client.

5.12 Actions

5.12.1 FDI Server – FDI Client interaction

FDI Clients invoke Actions by calling the InvokeAction method (see IEC 62769-5).

When an Action is invoked the FDI Server creates a state machine that is maintained while the Action is executing. The state may change in response to the builtin functions that are invoked by the Action, as well as in response to interactions with the FDI Client.

The FDI Server then creates a transient, non-browsable Variable in the Information Model for the exchange of information between the FDI Server and the FDI Client, henceforth referred to as the exchange variable. The NodeId of the exchange variable is returned to the FDI Client as an output argument of the InvokeAction method (see IEC 62769-5).

Once the state machine has been created, the exchange variables have been created, and the Action has started to execute, the InvokeAction method terminates, i.e. it does not remain active during the execution of the Action.

The FDI Server sends user interface requests to the FDI Client via the exchange variable, and the FDI Client sends user interface responses to the FDI Server via the exchange variable. The value of the exchange variable is an XML string (see IEC 62769-2). It contains the current state of the Action, as well as a user interface request or response.

The subscription service specified in IEC 62541-4 is used to allow the FDI Server to send user interface requests to the FDI Client via the exchange variable. The FDI Client subscribes to the exchange variable to receive user interface requests from the FDI Server. If a request is transitional the FDI Client may miss the request. The request will be held in the exchange variable until the FDI Client creates a subscription. Once the subscription is established, the FDI Server will respond with the current state and the pending request.

NOTE 1 TimeDelay with a short duration is an example of a transitional request.

The FDI Server can implement a server-defined time-out for user interface requests. Failure of the FDI Client to respond to a user interface request before the time-out expires can cause the FDI Server to abort the Action.

NOTE 2 The time-out is expected to be on the order of 20 min to 30 min similar to a session time-out of a web page.

The FDI Server retains the current state and the last request in the exchange variable even after the Action completes. The exchange variable retains its value until the FDI Client terminates the subscription.

An Action can be aborted by the FDI Client or by the Action itself.

The sequence diagram shown in Figure 18 shows the client/server interaction of an Action.

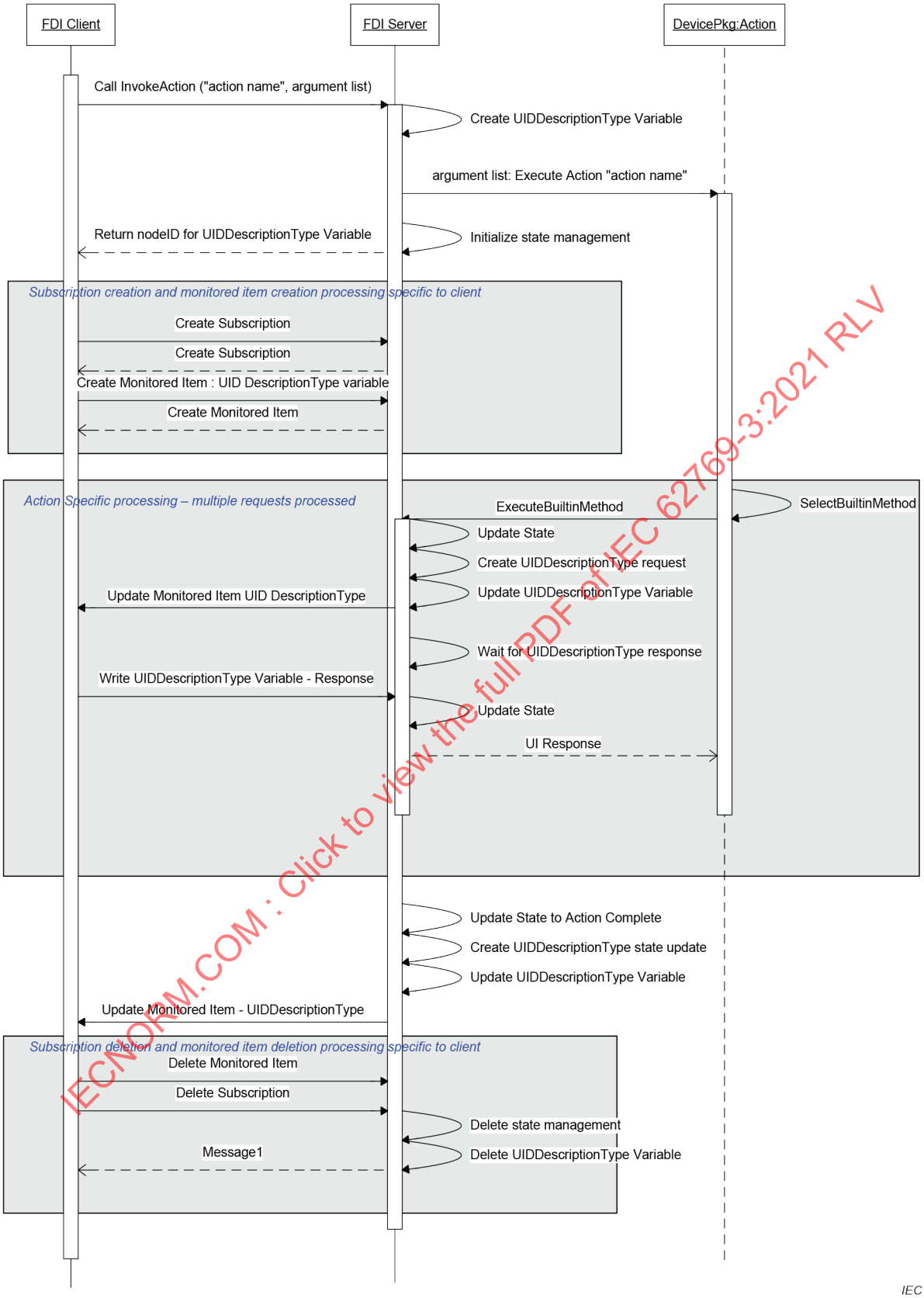

Figure 18 – Action execution

5.12.2 Action state machine

5.12.2.1 States

The Action state machine is shown in Figure 19.

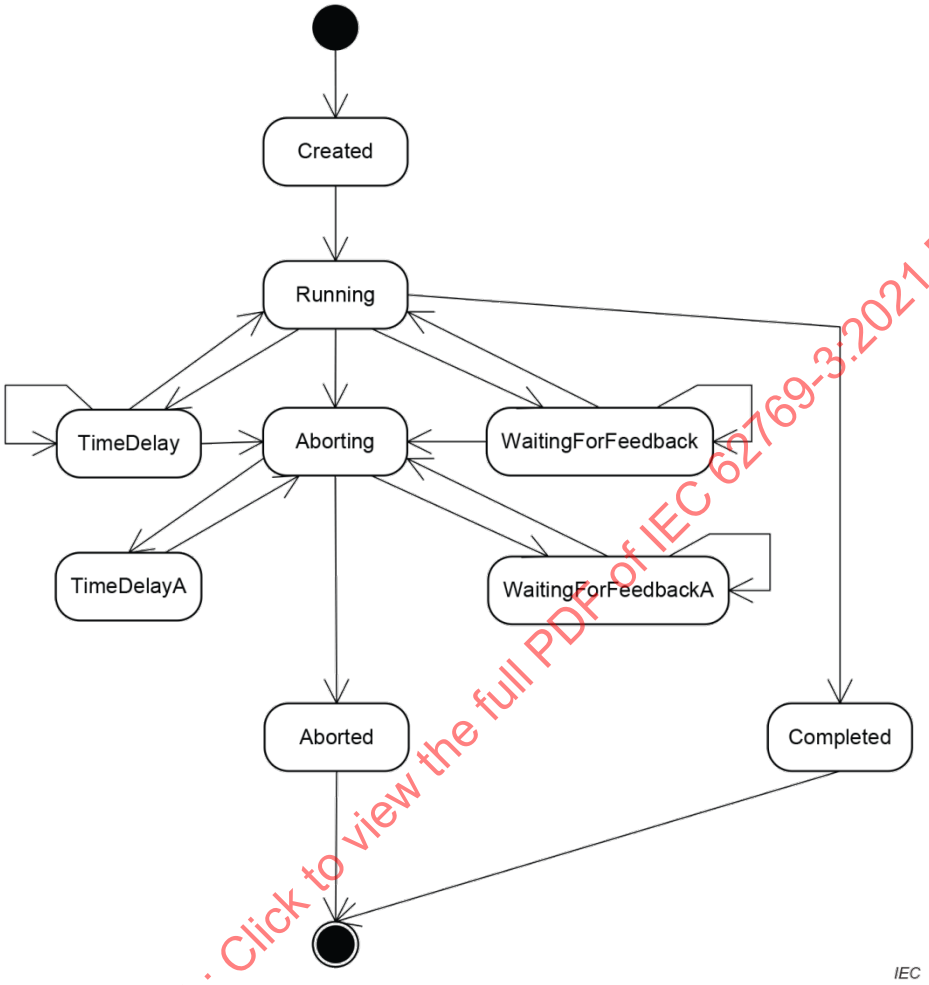

Figure 19 – Action state machine

The states of the Action state machine are specified in Table 1.

Table 1 – Action states

State	Description
Created	The initial state when the state machine instance is created by the FDI Server.
Running	The normal execution state.
TimeDelay	The state where the normal execution is suspended for a certain amount of time.
WaitingForFeedback	The state where the normal execution is suspended because a user interaction is needed.
Aborting	The state where the normal execution has been aborted and abort processing is carried out.
TimeDelayA	The state where the abort processing is suspended for a certain amount of time.
WaitingForFeedbackA	The state where the abort processing is suspended because a user interaction is needed.

State	Description
Completed	The state where the normal execution is completed.
Aborted	The state where the abort processing is completed.

5.12.2.2 State transitions

The state transitions of the Action state machine are defined in Table 2.

Table 2 – Action state transitions

Source state	Event	Destination state
Start State	FDI Server created a state machine for the Action.	Created
Created	Execution of the Action has started.	Running
Running	Execution of the Action has completed.	Completed
Running	Builtin function has been encountered that requires a time delay.	TimeDelay
Running	Builtin function has been encountered that requires user feedback.	WaitingForFeedback
Running	Abnormal termination of the Action has been initiated by either the FDI Server or the FDI Client.	Aborting
TimeDelay	FDI Server has decided to send time delay remaining to FDI Client.	TimeDelay
TimeDelay	FDI Server has calculated a new time delay to be sent to FDI Client.	TimeDelay
TimeDelay	Abnormal termination of the Action has been initiated by the FDI Client.	Aborting
TimeDelay	Delay time has expired.	Running
WaitingForFeedback	FDI Server has decided to send an updated feedback request to the FDI Client.	WaitingForFeedback
WaitingForFeedback	FDI Server has received feedback from FDI Client.	Running
WaitingForFeedback	Abnormal termination of the Action has been initiated by either the FDI Server or the FDI Client.	Aborting
WaitingForFeedback	FDI Server timeout period has expired with no response from FDI Client.	Aborting
Aborting	Builtin function has been encountered that requires a time delay.	TimeDelayA
Aborting	Builtin function has been encountered that requires user feedback.	WaitingForFeedbackA
Aborting	Execution of the Action has completed.	Aborted
TimeDelayA	Delay time has expired.	Aborting
WaitingForFeedbackA	FDI Server has decided to send an updated feedback request to the FDI Client.	WaitingForFeedbackA
WaitingForFeedbackA	FDI Server has received feedback from FDI Client.	Aborting
Aborted	FDI Server has destroyed the state machine for the Action.	Finish State
Completed	FDI Server has destroyed the state machine for the Action.	Finish State

5.12.3 Actions Proxies

EDD Actions specify EDD Methods that shall be executed at specific moments during the processing of variable values or during user interaction. In many cases, the FDI Server implicitly executes the EDD Actions, but in some specific cases, as specified in 5.12.4, the execution of EDD Actions is driven by the FDI Client.

In order to allow the FDI Client to drive the execution of EDD Actions, the FDI Server creates Actions Proxies. An Actions Proxy is an internal entity created by the FDI Server to encapsulate the EDD Methods specified in the EDD Action definition. An Actions Proxy thus corresponds to a single "*_ACTIONS" clause in EDDL and therefore to the entire set of EDD Methods specified in it.

The FDI Server assigns a name to the Actions Proxy. The Actions Proxy name is an unambiguous identifier, i.e. it uniquely identifies the Actions Proxy in the scope of a single device instance.

The FDI Server makes the Actions Proxy name available to the FDI Client via the XML descriptions associated to UID nodes (see 5.11.1). The FDI Client can thus drive the execution of an Actions Proxy when necessary by calling the InvokeAction method and passing the Actions Proxy name as argument.

NOTE The second argument of the InvokeAction method ("MethodArguments") is empty, since EDD Actions have no arguments.

When processing an InvokeAction call with an Actions Proxy name as argument, the FDI Server executes the entire set of EDD Methods associated to that Actions Proxy. As specified in a, the FDI Server executes those EDD Methods in the order they appear in the EDD Action definition, and if an EDD Method exits for an unplanned reason, the following EDD Methods are not executed.

Taking as reference the state transitions defined in Table 2, it means that:

- the state machine transitions from the state "Created" to the state "Running" when the execution of the first EDD Method in the Actions Proxy definition starts;
- in the meantime between the execution of two EDD Methods, the state machine remains in the state "Running";
- the state machine only transitions from the state "Running" to the state "Completed" when the execution of the last EDD Method in the Actions Proxy definition completes, or if any EDD Method exits for an unplanned reason.

All other state transitions remain the same.

5.12.4 Actions, EDD Actions and Actions Proxies

Actions are a provision of the FDI Server to allow FDI Clients to execute both EDD Methods in general and EDD Actions in particular. EDD Methods in general, with the exception of abort and action methods, are exposed in the Information Model as nodes under the ActionSet Object of the corresponding device or block node (see IEC 62769-5). EDD Actions on the other hand are not exposed in the Information Model. EDD Actions are made available to the FDI Client by putting the names of their corresponding Actions Proxies in the ListOfActions element in the XML description of the UID nodes (see 5.11.1).

The EDD Action types and the EDD constructs that use them are shown in Table 3 (see IEC 61804-3).

Table 3 – EDD Action types and the EDD constructs that use them

EDD Action type	EDD construct					UID
	VARIABLE	MENU	EDIT_DISPLAY	WAVEFORM	SOURCE	
Pre-read Actions	I	I				
Post-read Actions	I	I				
Pre-write Actions	I	I				
Post-write Actions	I	I				
Pre-edit Actions	E	E	E			E
Post-edit Actions	E	E	E			E
Init Actions		E		E	E	E
Exit Actions		E		E	E	E
Refresh Actions	I	E		E	E	E

As Table 3 indicates, in some cases, the FDI Server implicitly executes the EDD Actions (I), while, in other cases, the execution of EDD Actions is driven by the FDI Client, i.e. the FDI Client needs to explicitly start the execution of the EDD Actions in the FDI Server (E).

The FDI Server implicitly executes the following types of EDD Actions:

- Pre-read, post-read, pre-write and post-write actions, both for variables and menus;
- Refresh actions for variables.

Those types of actions shall not require user interaction; they are strictly intended to be used for Business Logic processing. Any action of one of those types that eventually requires user interaction will not perform the builtin but will return an error if possible.

The FDI Server implicitly handles pre-read, post-read and refresh actions for variables when the FDI Client reads online variables (see 5.7.3). Similarly, the FDI Server implicitly handles pre-write and post-write actions for variables when the FDI Client writes online variables, either in an immediate fashion (see 5.8.3) or in edit mode (see 5.8.4).

The pre-read, post-read, pre-write and post-write actions for menus are only used by the upload and download menus (see IEC 61804-4 and IEC 62769-2). The FDI Server implicitly handles pre-write and post-write actions for menus when the FDI Client transfers data to the device (5.2.2). Similarly, the FDI Server implicitly handles pre-read and post-read actions for menus when the FDI Client transfers data from the device (5.2.3).

The FDI Client explicitly starts the execution of the following types of EDD Actions:

- Pre-edit and post-edit actions for variables, menus, edit-displays and UIDs;
- Init, exit and refresh actions for menus, waveforms, sources and UIDs.

When those actions contain user interactions (see IEC 61804-4), they will require interaction between the FDI Server and the FDI Client. This is achieved by using Actions, as specified in 5.12.1. The FDI Client explicitly starts the execution of those types of actions in the FDI Server by calling the InvokeAction method and passing the name of the corresponding Actions Proxy as argument.

6 OPC UA services

6.1 OPC UA profiles

The set of services specified for OPC UA are grouped into standardized profiles as defined in IEC 62541-7. FDI Servers shall conform to the FDI Server Profile, which is specified as:

- including OPC UA "Standard Server";
- including OPC UA "DataAccess Server Facet";
- including OPC UA "Node Management Server Facet";
- including OPC UA "Method Server Facet";
- including OPC UA "Event Subscription Server Facet";
- including OPC UA "Auditing Server Facet";
- including FDI "FDI Information Model".

6.2 Service error information

6.2.1 Overview

FDI Servers provide service operations that are invoked through OPC UA services. Standard OPC UA service status information is returned by the FDI Server as a result of the service calls.

The OPC UA specification defines all services as having a standard response that includes a response header containing general and service-specific response codes in accordance with IEC 62541-4. The response code structure contains diagnostic information that returns both an error code as well as localized text for the error. FDI Servers shall fill in the diagnostic records including localized text for the reported errors.

The OPC UA diagnostic record allows Servers to include "inner" status information. FDI Servers will provide technology binding specific errors in the "inner" status record.

6.2.2 OPC UA services and their response

When the FDI Client submits a Service request Message to the FDI Server, if the service is supported and executed, the FDI Server generates a success/failure code that it includes in a positive response Message along with any data that is to be returned. Each Service request has a RequestHeader and each Service response has a ResponseHeader.

The ResponseHeader is a structure that has data members used to convey EDDL diagnostics information, the serviceResult and the diagnosticInfo.

The serviceResult is the standard, OPC UA-defined result of the Service invocation. The serviceResult type is StatusCode, which defines a standard numerical value that is used to report the outcome of an operation performed by an FDI Server. This code may have associated diagnostic information that describes the status in more detail.

The diagnosticInfo is a structure that is intended to return vendor-specific diagnostic information.

6.2.3 Mappings of EDDL response codes to OPC UA service response

When FDI Clients use OPC UA services to read and write the Attributes of Parameters, they receive back as part of the FDI Server response a ResponseHeader with success/failure code and diagnostics information.

The FDI Server uses the serviceResult and the diagnosticInfo data members of the ResponseHeader to return error and diagnostics information related to failure of execution of EDDL variable actions, including PRE_READ, POST_READ, PRE_WRITE and POST_WRITE actions.

The StatusCode returned in the serviceResult data member of the ResponseHeader is also used to handle the EDDL VALIDITY attribute. Any attempt to access an invalid variable will be reported to the FDI Client as the result of a service call. The service returns a "Bad Failure" in the Severity bit of the StatusCode. In addition, the diagnosticInfo data member of the ResponseHeader can be used to provide detailed diagnostics on the failure. The FDI Server shall also deal with the fact that VALIDITY can be the result of the evaluation of a conditional expression. In that case, FDI Clients rely on the FDI Server notification capabilities when the model dynamically changes owing to a conditional evaluation.

The serviceResult and the diagnosticInfo data members of the ResponseHeader are used to return error and diagnostics information related to EDDL response codes. EDDL response codes specify values that a device may return as the result of an operation. Each EDDL variable, record or value array can define its own associated set of response codes.

The serviceResult is used to return a status that corresponds to the EDDL response code TYPE attribute. The Severity bits of the StatusCode are set based on the response code TYPE according to Table 4.

Table 4 – OPC UA severity bits and EDDL response codes TYPE

OPC UA Severity	EDDL Response Codes Type
Good Success	SUCCESS
Uncertain Warning	MISC_WARNING, DATA_ENTRY_WARNING
Bad Failure	DATA_ENTRY_ERROR, MODE_ERROR, PROCESS_ERROR, MISC_ERROR

The symbolicIdIndex, localizedTextIndex and the additionalInfo data members of the diagnosticInfo are used to return the response code and the text description gotten from EDDL response codes definitions. It is the FDI Server's responsibility to translate the integer response code into its corresponding text description and fill in the diagnosticInfo.

The symbolicIdIndex data member is used to return the numeric response code from the EDDL RESPONSE_CODE. The numeric code shall be converted into a string in the stringTable. The symbolicIdIndex contains the index into the stringTable.

The localizedTextIndex data member is used to return the DESCRIPTION attribute from the EDDL RESPONSE_CODE. The DESCRIPTION string is conveyed to the FDI Client in the stringTable data member of the ResponseHeader parameter. The localizedTextIndex contains the index into the stringTable.

The additionalInfo data member is used to return the HELP attribute from the EDDL RESPONSE_CODE.

In addition to the response codes described via EDDL, standard response codes defined by the underlying communication protocols may also be returned.

6.3 Parameter value update during write service request

The FDI Server maintains an Information Model that contains Online variables that cache the value of the device variables. The specified behavior for OPC UA is for the Server to only store in the Online Variables those values that the Server has read from the device.

The FDI Server is not allowed, according to OPC UA specified behavior, to write a value both to the device and to the Online variable.

6.4 Localization

The Information Model defined for FDI, IEC 62769-5, is based on OPC UA. The OPC UA specification defines descriptive attributes and properties of elements to be localized strings. The FDI Server provides localized information for the OPC UA specified localized attributes and properties in the Device Type and Device Instance nodes.

The FDI Server uses information provided by the descriptive component in the FDI Package for a device type to create localized strings. FDI Packages support the specification of localized strings for descriptive information. If the device vendor provides such information, the FDI Server uses the appropriate localized string as the value for device type attributes and properties when responding to an FDI Client.

If the descriptive element in the FDI Package does not provide localized information, either no information or no information for the requested locale, the FDI Server will return the English default string.

Multiple clients connecting at the same time may eventually request the FDI Server to return localized attributes and properties in different languages. The FDI Server shall support multiple languages simultaneously when the clients requesting different languages connect to different Device Type or Device Instance nodes.

When clients requesting different languages connect at the same time to the same Device Type or Device Instance node, the support to multiple languages is optional. If the FDI Server does not support multiple languages in that situation, then it shall implement a "first wins" solution, i.e. it will use the language requested by the first client that connected to the Device Type or Device Instance node when returning localized attributes and properties to all subsequent clients.

6.5 Audit events

FDI Servers shall provide support for vendor-specific audit trail functionality. The support for auditing in the FDI Server is specified in IEC 62769-5.

7 Communication

7.1 Notation

Clause 7, describing communication, contains diagrams showing the Information Model, IEC 62769-5. The notation used in these figures uses the standards defined by OPC UA. These standards are summarized in IEC 62769-5.

7.2 General

7.2.1 Concepts

The FDI Server is responsible for managing communications. The FDI Server can support three types of communication infrastructure components:

- System communication hardware (see Figure 1);
- FDI Communication Server;
- Communication gateways (Nested Communication).

The FDI Server can support system-specific communication hardware (see Figure 1). The FDI Server interacts with the driver of the system communication hardware through proprietary interfaces to process fieldbus communication. System-specific communication management is not in the FDI specification's scope.

Figure 20 shows a possible architecture example of system communication integration.

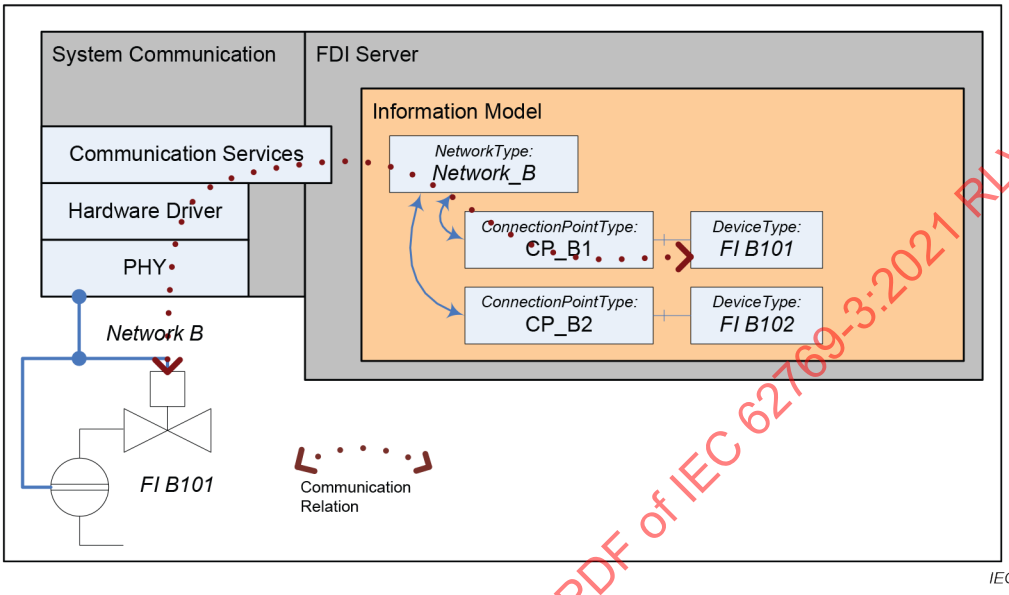

Figure 20 – System communication integration example

The FDI Server can implement access to physical networks through FDI Communication Servers (see Figure 1) (IEC 62769-7). The FDI Communication Server implements the access to the physical network. The interface between the FDI Server and the FDI Communication Server is based on OPC UA. The FDI Communication Server implements the OPC UA Server function. The FDI Server implements the OPC UA Client function. The FDI Communication Server implemented Information Model enables the access to the communication services.

An FDI Communication Server integration example is shown in Figure 21.

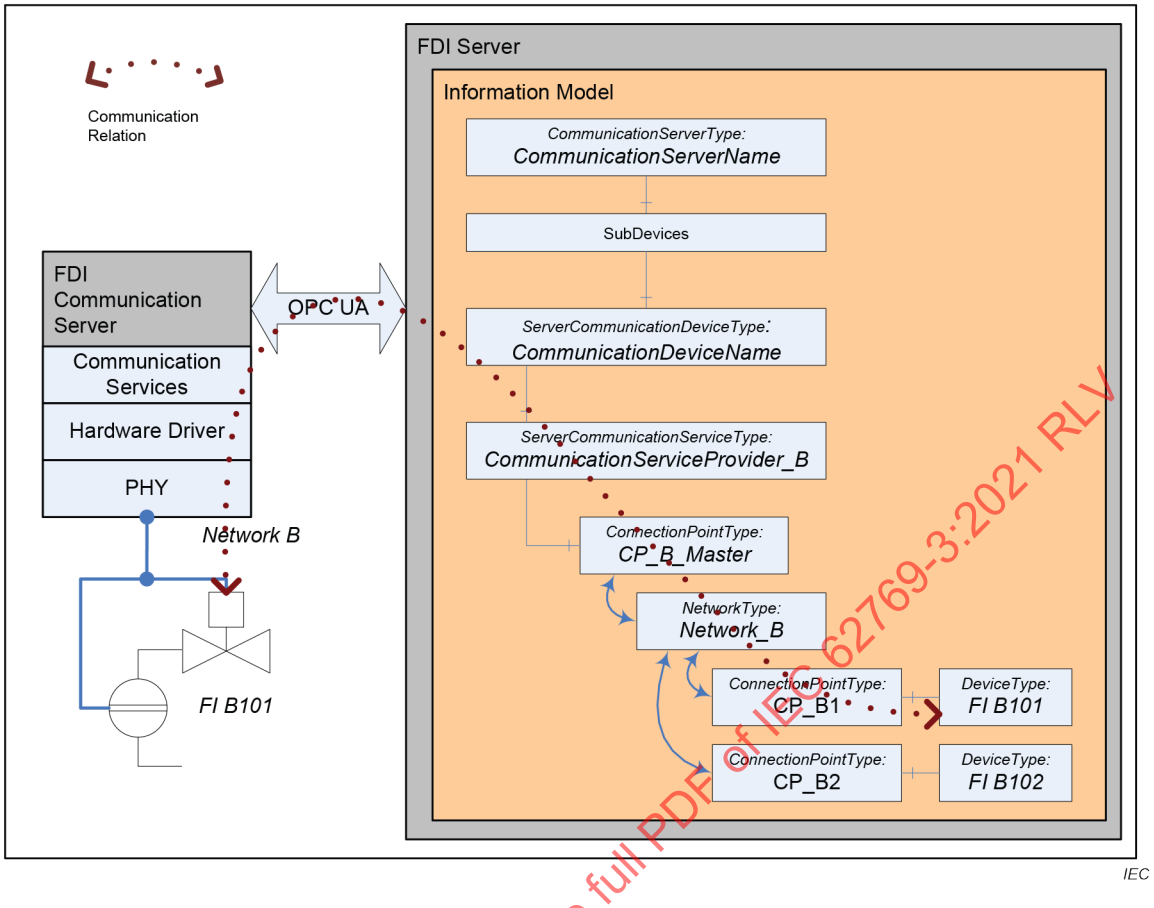

Figure 21 – FDI Communication Server integration example

In terms of the Information Model structure, the system communication hardware and the FDI Communication Server represent root communication devices.

FDI supports Nested Communication. The term "Nested Communication" stands for the ability of a system to process communication across protocol boundaries in heterogeneous networks. The insertion of additional communication gateway devices into the topology as shown in Figure 22 enables the handling of heterogeneous networks. These communication gateway devices implement the bridging functionally between different networks (gateway firmware). The gateway firmware implemented bridging functionality is also implemented in the Business Logic provided with the FDI Package describing the communication gateway. The FDI Server interacts only with this Business Logic of the communication gateway to process the Nested Communication function.

A communication gateway integration example is shown in Figure 22.

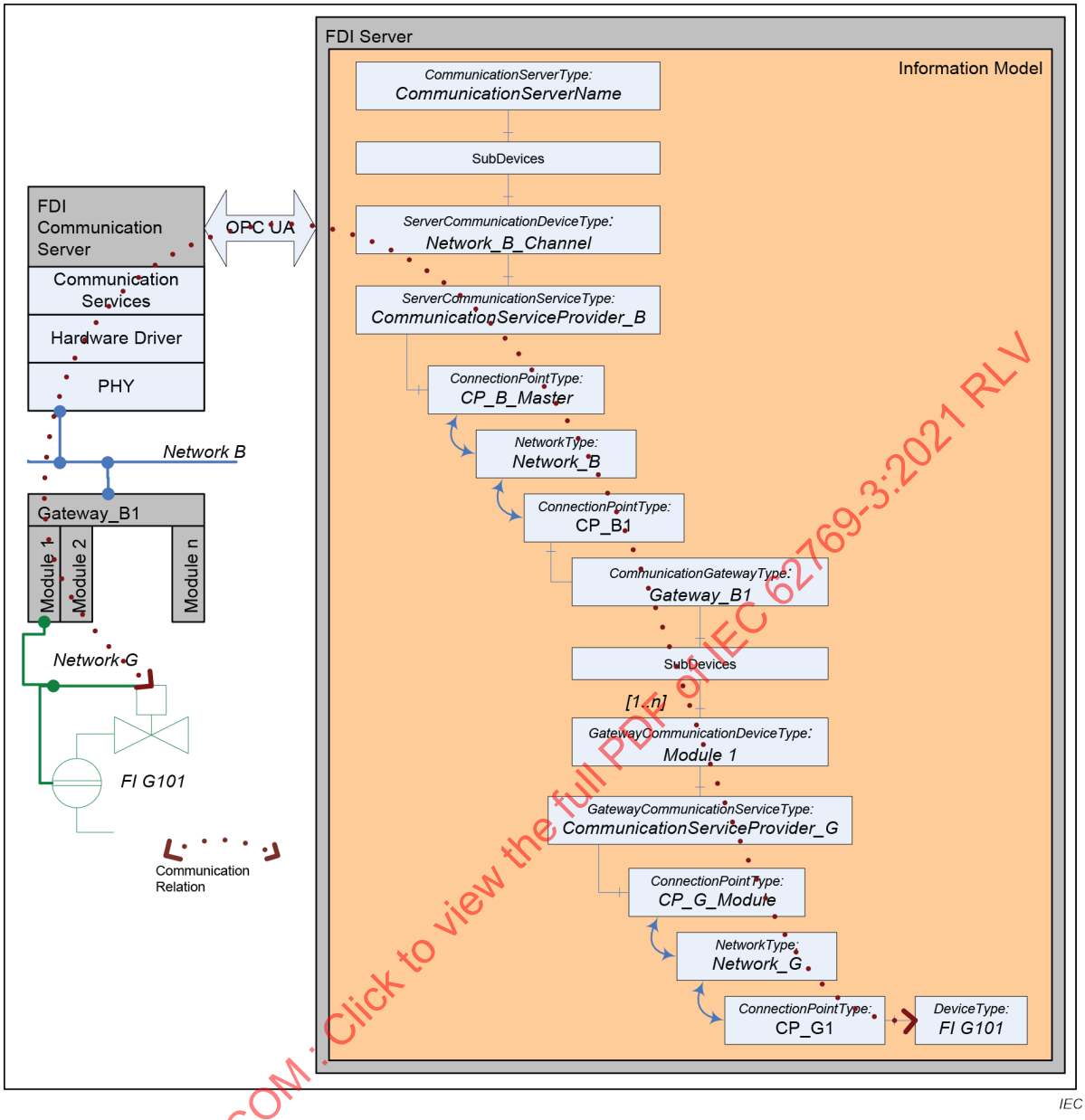

Figure 22 – Gateway integration example

7.2.2 Terms

The following contains a list of terms used in 7.3:

1) A Connection Point is an instance of a ConnectionPointType (see IEC 62769-5).
2) A Device is an instance of a DeviceType.
3) A Connection Point associated to a Device is called Device Connection Point.
4) A Connection Point associated to a Communication Device is called Communication Device Connection Point.
5) A Network is an instance of NetworkType (see IEC 62769-5).
6) FDI Communication Server (see IEC 62769-7).

7.3 Communication Service processing

7.3.1 Communication Service invocation

IEC 62769-7 specifies that both Gateways and FDI Communication Servers implement the communication services. Those services are specified in IEC 62769-7.

In order to allow the flexibility that is necessary to represent a variety of different scenarios involving communication between the communication server and the physical devices, IEC 62769-7 specifies that some communication services are provided through a communication service provider. While the communication service provider allows multitasking and enhances the communication capabilities of a communication device, it requires parallel execution in the server. The details about communication service providers (ServerCommunicationServiceType and GatewayCommunicationServiceType) are specified in IEC 62769-7. The requirements for parallel execution in the service are described through the rules stated in Clause 8.

The difference between a Gateway and an FDI Communication Server is about how or where these services shall be invoked:

a) If the communication device is an FDI Communication Server, the FDI Server invokes a communication service directly at the FDI Communication Server using an OPC UA Method service Call specified in IEC 62541-4. This call will end up in actual fieldbus communication.

b) If the communication device is part of a Gateway, the FDI Server invokes the communication service in terms of invoking an Action implemented by the Business Logic of the specific Gateway. The behavior (reaction) of the Gateway Business Logic is described in IEC 62769-7.

7.3.2 Analyze communication path

The Information Model defined in IEC 62769-5 supports a hierarchical topology. As shown in Figure 22, the topology reflects the physical network topology. The communication path analysis function allows the FDI Server to determine how communication messages need to be propagated from the Device that triggered a communication request to the Communication Device implementing the network access (root communication device) and which communication relations need to be activated before. Subsequent text will only consider the root communication device based on the FDI Communication Server.

The FDI Server identifies the communication path between a Device and an FDI Communication Server according to the following rules:

a) Topology iteration starts from the node representing the Device passing the elements Device Connection Point, Network, Communication Device Connection Point to the Communication Device within the same Network hierarchy. In this way, the FDI Server determines the local communication relation. A Communication Device that is associated next to the Device implements the communication service provider for this Device, this means the FDI Server shall propagate the communication service request between the Device and Communication Service Provider.

b) The FDI Server identifies a communication gateway along its Information Model structure as demonstrated in Figure 22. The key indicators are the Communication Device organized below a Device using the "has Component" relation. This specific device is called Gateway Head Station, which is connected to a different Network via a Connection Point. From here, the iteration continues as described in a).

c) The topology iteration procedure ends with finding the communication root device. The FDI Server identifies an FDI Communication Server (communication root device) because it has no association to other networks than the network for which the FDI Communication Server implements the communication service provider.

NOTE How the FDI Server determines System Communication Device is out of the scope of this document.

7.3.3 Manage communication relations

Prior to any data exchange related transfers, the FDI Server needs to establish or activate the communication relation between the Device representations in the Information Model and the physical network connected device. The invocation sequence of the communication service Connect on any of the Communication Devices along the communication path shall begin with the root communication device. The communication service Connect is specified in IEC 62769-7.

The Device Connection Point contains the address information to be used for the Connect service. The Information Model element FunctionalGroupType:Identification contains optionally required protocol-specific device type identification data. The Connect service argument names shall match with the browse names of the Information Model elements that hold related values (browse name matching).

The successful execution of service Connect activates the local communication relation between a Device and a Communication Device associated to the same network. The successful execution of service Connect on a network higher in a hierarchy is a prerequisite to a successful execution of the service Connect on a network lower in the hierarchy. The reason for this is the Gateway Business Logic that can invoke other communication services requiring an activated communication relation.

The FDI Server manages a CommunicationRelationId in accordance with IEC 62769-7.

A connection abort indication or the invocation of service Disconnect as described in IEC 62769-7 deactivates the local communication relation and any of the local communication relations in networks lower in the hierarchy.

7.3.4 Communication service request mapping

The FDI Server receives communication service requests from Devices or Gateways through:

a) the Online Variable Read;

b) the Online Variable Write;

c) the Business Logic invoking the communication related EDDL Builtin function, for example, send, send_all_values, send_command, send_command_trans, send_trans, send_value, WRITE_COMMAND, READ_COMMAND, and so on.

Like the Device, all of these communication service requests related source events apply to the EDDL PROFILE (IEC 61804-3). The FDI Server shall handle the communication service requests in accordance with EDDL-defined PROFILEs.

Since the EDDs shipped with the FDI Packages can describe relations between VARIABLE elements and COMMAND elements, the Variable Read or Write access can be mapped to a COMMAND description because of a particular COMMAND description that refers to a particular VARIABLE. Such COMMAND descriptions contain communication service arguments and instructions about how to create the data payload of a communication service.

If no COMMAND Description is present, the VARIABLE identifier (Name) and the VARIABLE value are the only communication service Transfer arguments.

Once the FDI Server has determined the communication service arguments from EDD, it can map it to the communication service Transfer (IEC 62769-7) arguments based on name matching. Transfer arguments shall have the same names, data types and semantics as described for a protocol-specific COMMAND definition.

EXAMPLE The COMMAND description contains the attributes SLOT, INDEX, REQUEST, REPLY. The protocol-specific signature of the Transfer service shall envision:

```
Transfer(
  [in]  String              communicationRelationId,
  [out] Integer             serviceError,
  [in]  unsigned char       SLOT,
  [in]  unsigned char       INDEX,
  [in]  char[]              REQUEST,
  [out] char[]              REPLY)
```

NOTE The arguments `communicationRelationId` and `serviceError` are described in IEC 62769-7.

7.3.5 Communication service request propagation

Subclause 7.3.5 describes how the FDI Server manages the communication message propagation along the communication path. The following Figure 23 represents an example scenario derived from Figure 22.

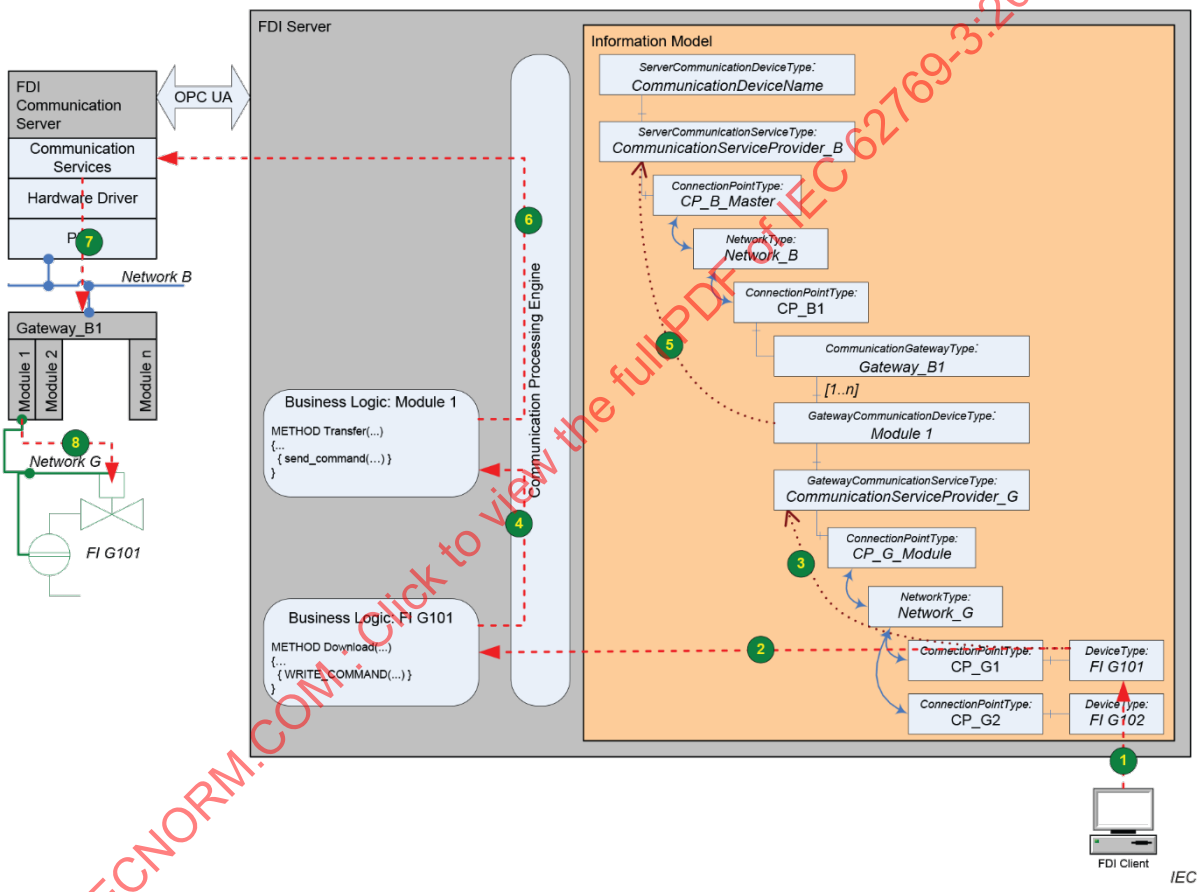

Figure 23 – Message propagation example scenario

NOTE 1 The numbers in brackets (1) to (8) used in the the following description refer to elements in Figure 23.

The FDI Server detects a communication request because of an FDI Client (1) invoking an Action (2). The processing of that Action (METHOD Download) invokes the communication request-related EDDL Builtin such as WRITE_COMMAND that has been mapped to Transfer service arguments, as described in 7.3.4.

The FDI Server processes the message propagation related iterations upwards through the hierarchy of the topology.

The FDI Server shall always determine the next Communication Service Provider along the communication path (3) and (5) (see 7.3.2) and invoke the service Transfer there.

If the Communication Service Provider processing the Transfer service is in an FDI Communication Server, the Transfer service needs to be invoked using the OPC UA service Call (6).

NOTE 2 The Communication Service Provider in the FDI Communication Server sends the protocol-specific message to the physical network (7).

If the Communication Service Provider processing the Transfer service is in a Gateway, the processing of that related Business Logic will cause other communication request-related EDDL Builtin invocations, for example, send_command (4). The iteration procedure enters the next recursion determining the next Communication Service Provider along the communication path (5) (see 7.3.2) and invokes the service Transfer there.

NOTE 3 The gateway implementation of service Transfer is device-specific. The Transfer logic wraps the incoming Transfer argument values and creates another message to be sent out calling a communication request related EDDL Builtin (see IEC 61804-5).

The Transfer logic can invoke multiple communication requests as this might be needed to manage the protocol bridge function. The physical gateway device unwraps and forwards the message (4) to the physical device (8).

The FDI Server managed communication propagation process is a recursive process in which the Business Logic execution of one Device can invoke the Business Logic execution of a different Device. This FDI Server needs to maintain an invocation stack.

7.3.6 Communication error handling

The FDI Server is responsible for handling communication errors. The FDI Server detects errors either from the Communication Service Provider returned service invocation results as specified in IEC 62769-7 or through EDDL builtins for abort processing.

The FDI Server aborts all communication Actions waiting for a response if a communication error or abort is received.

The FDI Server will return a failure to the originating service if a communication error or abort is received.

7.4 FDI Communication Server specific handling

7.4.1 Discovery

IEC 62769-7 describes the FDI Communication Server implemented discovery support in terms of:

a) VARIABLE definitions describing the FDI Communication Server's identification data that are represented in the FDI Server hosted Information Model;

b) FDI Communication Server implemented usage of IEC 62541-4 specified discovery services.

The FDI Server uses the services FindServers and the GetEndpoint IEC 62541-4 specified discovery service set to determine the FDI Communication Server. The FDI Server shall match the FDI Communication Server's defined identification data with values returned from the functions FindServers and GetEndpoints.

The FDI Server implements the IEC 62541-4 specified Discovery Server.

7.4.2 Information Model synchronization

In accordance with IEC 62769-7, the FDI Communication Package contains an EDD element describing the VARIABLES and Business Logic mapped into the Information Model. IEC 62769-7 also describes the overlap between the FDI Communication Server-hosted Information Model and the FDI Server-hosted Information Model. This overlap represents the shared Information Model.

The FDI Server synchronizes the shared Information Model:

a) Any access to an offline node of the Information Model is locally handled through the Information Model.

b) The FDI Server handles a write access to an online node of the Information Model by performing the same write access in FDI Communication Server-hosted Information Model.

c) Any read of an online node of the Information Model results in a read operation on the corresponding node of the FDI Communication Server-hosted Information Model.

d) Any configuration changes affecting the modular structure represented in the Information Model are copied from the FDI Server-hosted Information Model to the FDI Communication Server-hosted Information Model.

8 Parallel Execution within the FDI Server

8.1 Motivation

Within the EDDL concept, each device is described by a set of parameters and an EDD that describes and handles relations between the parameters and their attributes. Each combination of the EDD and the respective set of parameters builds an entity describing a device instance (called EDD Entity hereafter). EDDL allows only synchronous operation with such an entity without any parallel execution. Therefore, when the EDD Entity is performing an action (e.g. reading a variable value, executing a method, etc.), any other action request shall be postponed until the action execution is finished.

As long as there is no relation between EDD Entities the action for EDD Entities can be executed independently and subsequent action requests can be queued without any risk to force a deadlock.

As soon as relations between EDD Entities have to be handled, the FDI Server has to control the execution within the EDD Entities in a way that deadlock scenarios or parallel execution within one EDD Entity are prohibited.

Nested communication is a concept based on interaction between EDD Entities. When handling multiple communications at once, the FDI Server has to handle actions in the FDI system in parallel. To prevent deadlock scenarios, the FDI Server has to follow well-defined execution rules.

8.2 Internal structure of the EDD interpreter

A core component of an FDI Server is the EDD interpreter (see Clause A.1). The EDD interpreter can be seen as a component that consists of EDD entities. Each EDD entity itself consists of the following parts:

- An associated EDD for a device or a component of a modular device.
- A set of data representing the state of the EDD entity and containing data that the interpreter requires to run the EDD associated with the data set. This data for example might contain the offline data set for the associated device and cached data of the connected device. It also contains additional information the interpreter needs for EDD-specific calculations.

- The interpreter logic that is triggered from outside and when triggered interprets the EDD, performing subsequent activities, changes the state and delivers calculation results.

8.3 Rules for running an EDD entity

As mentioned in 8.2, an activity at EDD entities is initiated always by a trigger. There are two kinds of triggers:

a) A trigger from the EDD entity itself that the interpreter logic requires for a correct EDD execution. An example for such triggers is periodic updates of dynamic variables.

b) A trigger that is a consequence of a service request from outside the FDI Server.

   For example:

   1) service requests from an FDI Client;
   2) service requests from an OPC-UA client.

The execution of an activity at an EDD entity shall follow the rules given below:

c) An activity at an EDD entity is always a consequence of a trigger.

d) An activity at an EDD entity cannot be interrupted.

e) An activity runs until the activity is finished or aborted.

f) An EDD entity can only execute one activity at a time.

g) An activity executed by an EDD entity always performs a non-interruptible process from the perspective of the EDDL logic. Such processes are for example:

   1) calculation of EDD objects,
   2) reading a variable from a device,
   3) writing a variable to a device,
   4) editing a variable,
   5) any activity that is embraced with pre- and post actions,
   6) executing an EDD method.

h) An active EDD entity may initiate sub-activities. While a sub-activity is ongoing the current activity at the EDD entity is paused. There are two kinds of sub-activities:

   1) the active EDD entity calls another EDD entity (e.g. nested communication or calls using cross-block and cross-module references),
   2) the active EDD entity requests another external service (e.g. communication, request for user interaction).

i) While an activity is paused re-entrance for activities is possible for those activities that are a consequence of the paused activity. Activities started by other triggers have to be blocked until the paused activity at the entity is finished.

j) When an activity chain of a trigger is blocked, there exists a blocking relation to another trigger and its activity chain. If this activity chain is blocked too, there is again a blocking relation to another trigger. Such series of blocking relations have to be monitored for recursion each time an activity chain has to be blocked. Recursion in the series of blocking dependencies indicates a deadlock scenario.

k) The server can resolve deadlock scenarios by aborting one of the involved activity chains.

For illustration, some examples on how to run an EDD entity in the FDI Server are described in Annex C.

Annex A
(informative)

FDI Server functional structure

A.1 FDI functional elements

The normative definition of an FDI Server is as shown in Figure 1. A non-normative view of an FDI Server shows the functional components that comprise the FDI Server and is shown in Figure A.1.

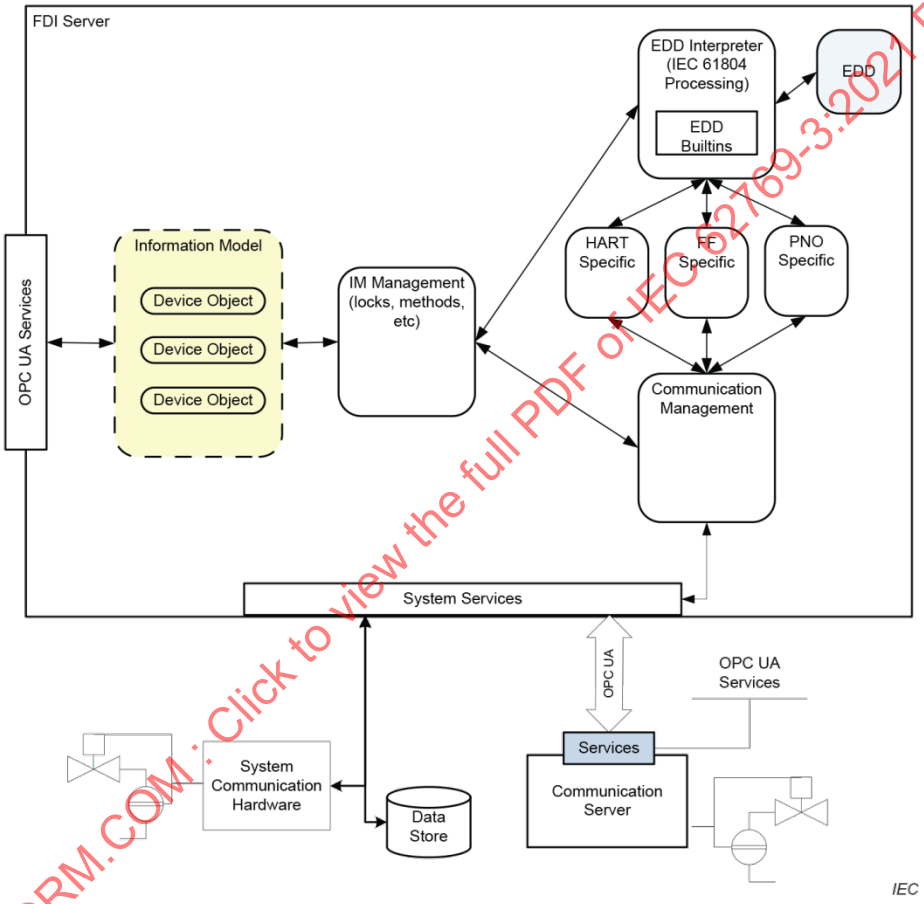

Figure A.1 – Functional components of an FDI Server

The FDI Server functionally contains an EDD interpreter that conforms to IEC 61804-3. The EDD interpreter provides descriptive information that is exposed in the Information Model, for example, device variables and standard menus. The EDD interpreter also provides the method execution functionality for IEC 61804-3.

Although IEC 61804-3 defines a standard EDD language, there are protocol-specific differences between EDDs. The FDI Server implements protocol-specific components. The protocol-specific components are used both as part of the EDD interpretation as well as for formatting communication.

The FDI Server contains node management functionality that provides the support for the Information Model. The Information Management component maintains the Information Model, handles multi-user requests, including lock management, and executes methods. The functionality provided by the Information Model management component is not restricted to FDI functionality. The IM management component may also provide general OPC UA functionality unrelated to FDI.

Service requests that result in physical read and write operations are passed to a communication manager. The communication manager provides the functionality required for communication, including state management of the communication requests. The communication manager contains the information to interact with system communication devices.

The communication manager interacts with protocol-specific components to create the actual messages transmitted on the fieldbus. The protocol-specific components interact with the EDD interpreter to retrieve information from the EDD to create the protocol-specific messages. The messages may be commands as in HART or the messages may be service requests as in FF. The actual message is created by the protocol-specific component.

The communication manager is responsible for managing the Nested Communication. The communication manager initiates the communication chain through the creation of protocol-specific messages. The communication manager then passes the message through the chain of communication devices in the topology until a top-level device is reached. The communication manager then interacts with the communication drivers for the top-level device. The interaction may be proprietary if the communication is through a system device. The communication may also be standardized through OPC UA to an FDI Communication Server.

A.2 FDI Server extension

An FDI Server can be extended to support future descriptive and protocol technologies through the addition of new interpreters and protocol handlers as illustrated in Figure A.2.

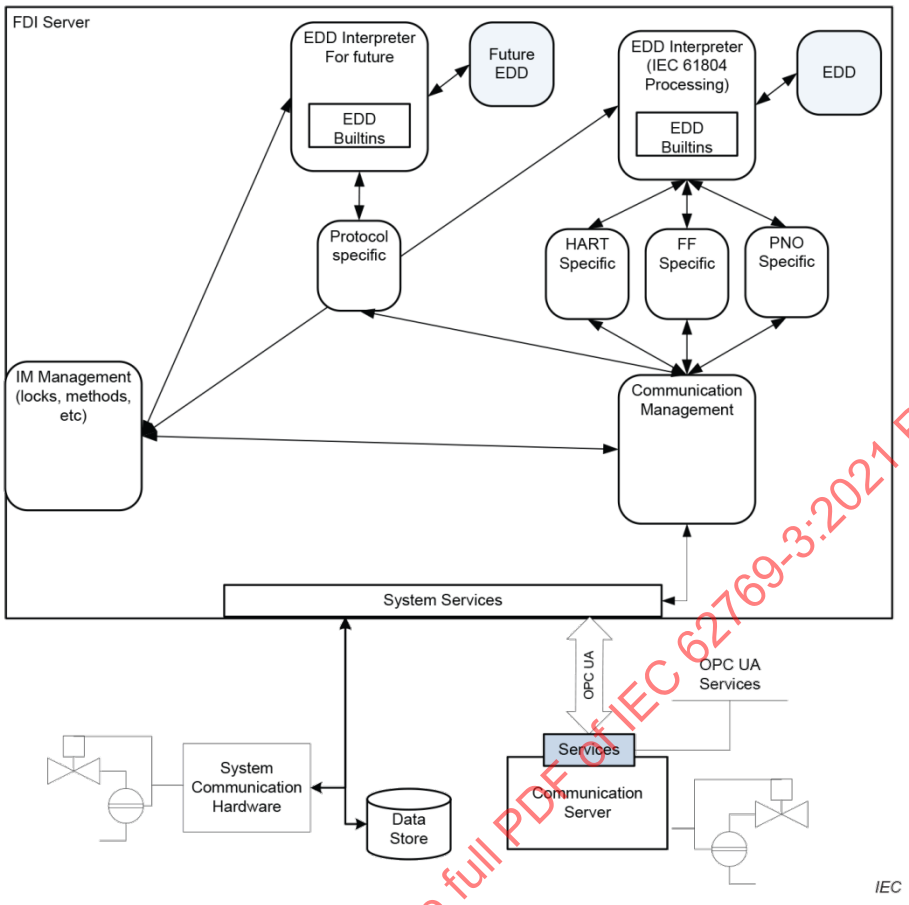

Figure A.2 – FDI Server extensions

These extensions, just like the support for FF, HART, and PNO, are specific to the implementation provided by the vendor of the FDI Server.

Annex B
(informative)

Access privileges and user roles

B.1 User roles and usage case

The specification for the Device Definition of an FDI Package contains a CLASS attribute in some elements. The CLASS attribute is supplied by the device vendor in the FDI Package and defines the intended usage of the Information Model element. The FDI Server makes the usage categories available to the system so user access and visibility to Information Model elements can be controlled by the system, possibly through the enforcement of system policies. The FDI Server reacts to the system defined policies to independently enforce the read/write access and public/private visibility of Information Model elements made available to users.

Figure B.1 depicts the relationship between the FDI Package, FDI Server, FDI Client, system, and user.

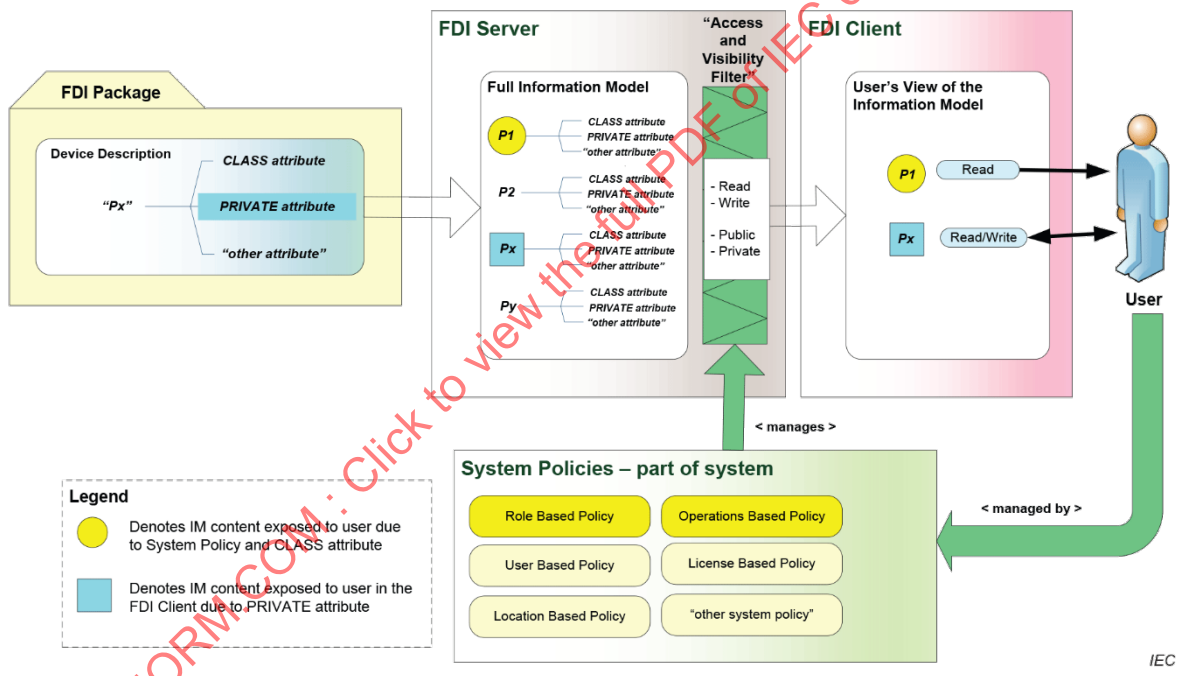

Figure B.1 – User roles and access privileges

The following relationships are noted:

The FDI Device Package identifies attributes associated with each element in the device description. These attributes become part of the Information Model that is managed by the FDI Server.

The CLASS attribute is made available to the system through the Information Model. It is an attribute for identifying the use cases, or usage scenarios, that are applicable to the Information Model element. The FDI host can use the CLASS attribute – in particular the value SPECIALIST – to determine whether the user of the FDI Client has access to the Information Model element and what level of access is allowed. The mechanism for making this determination is part of the system policy model that is internal to the system and outside of the scope of the FDI.

The system shall convey the access level for each Information Model element to the FDI Server so it can enforce the access model of the Information Model element for the user. The FDI Server is only the enforcer of the access rules, it does not decide "who" or "why". The system makes all of the "who" and "why" decisions, using both the CLASS attribute provided in the FDI Package and the system policies managed by the system. Some potential system policies may include, but are not limited to the following:

- Role-Based Policy,
- User-Based Policy,
- Location-Based Policy,
- Operations-Based Policy,
- License-Based Policy,
- Other Policies.

B.2 Private data usage

Among the attributes that have been prescribed by the FDI specification is an attribute for identifying whether an Information Model element, including data and Actions, is private to the elements in the FDI Device Package. This attribute, referred to as PRIVATE attribute in Figure B.1, determines whether an element in the Information Model is visible to FDI Clients during browse operations on the Information Model. The FDI Package elements have prior knowledge of private data and Actions in the Information Model and are able to access these private elements in accordance with access rules defined by CLASS attribute and system policy.

Access and visibility are independent attributes. For example, a private Action may be limited to user access during online usage scenarios.

System policy shall not be allowed to override the PRIVATE attribute in the Information Model. For example, the system policy cannot make private data public or public data private. The PRIVATE attribute is internally managed by device vendors.

Annex C
(informative)

Parallel execution within the FDI Server – Examples

C.1 Simple example for a synchronous execution

The generic examples in Clause C.1 are intended to visualize, for a better clarification, the rules given in 8.3.

Figure C.1 schematically shows the simplest example of a synchronous execution of two triggered activities. Both activities can be executed independently, because different EDD entities are involved.

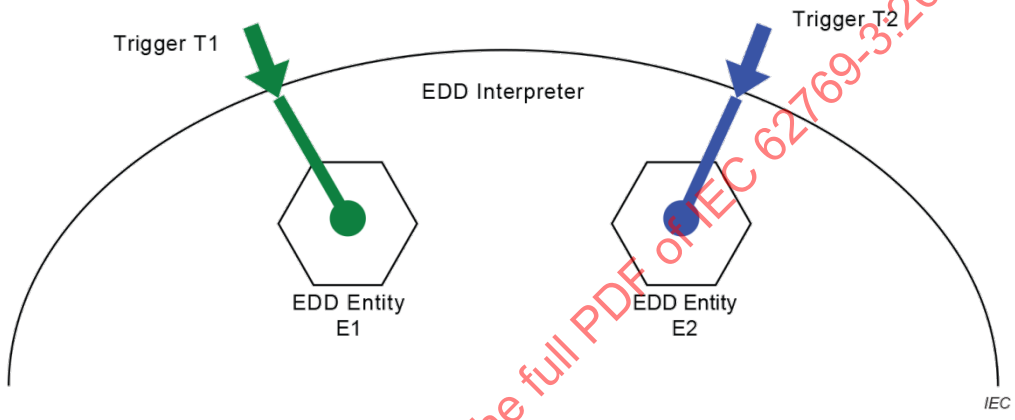

Figure C.1 – Synchronous execution of two triggers

C.2 Example for a concurrent execution

Figure C.2 shows a use case where two synchronous triggers try to access one and the same activity. While the activity of trigger T1 is executed, trigger T2 is blocked.

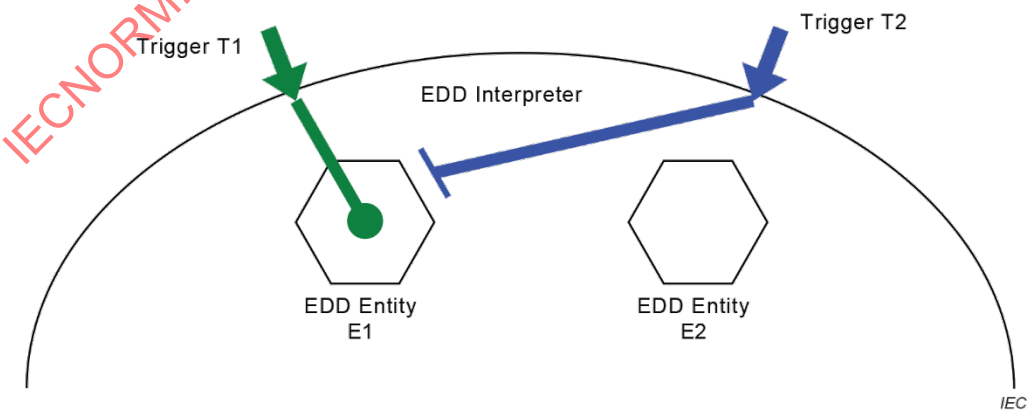

Figure C.2 – Concurrent execution of two triggers (step 1)

In Figure C.3 the activity of T1 is paused in EDD Entity E1 to execute a sub activity in E2. The activity of T2 is blocked until the activity of T1 is finished.

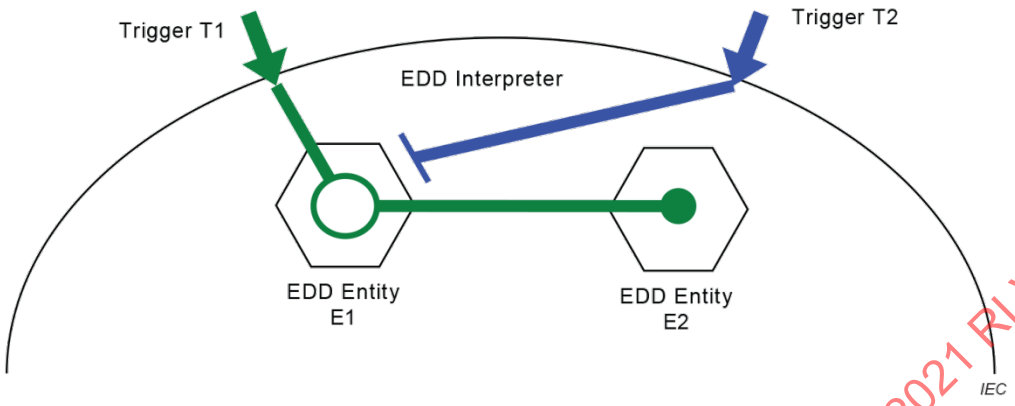

Figure C.3 – Concurrent execution of two triggers (step 2)

In Figure C.4, the sub activity in E2 initiates another sub activity back again in E1. This call back is allowed, because it is a direct consequence within the activity chain initiated by trigger T1 while activities of trigger T2 continue to be blocked.

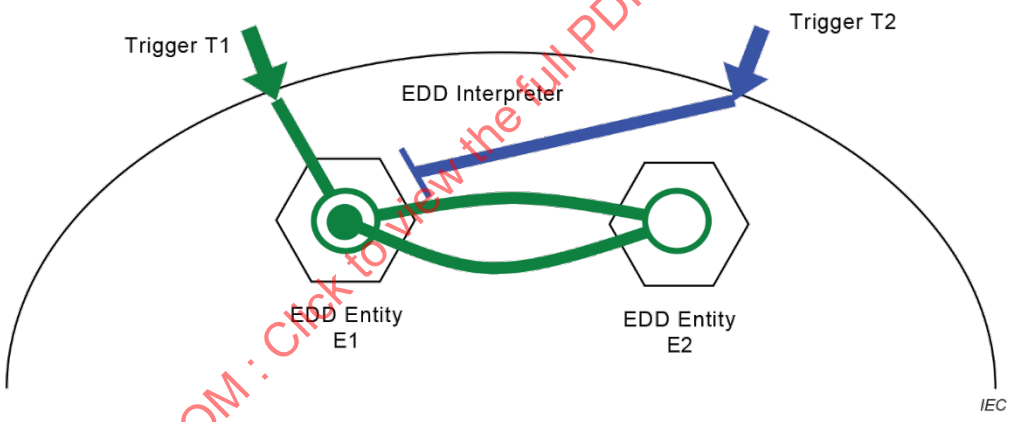

Figure C.4 – Concurrent execution of two triggers (step 3)

Figure C.5 shows that the activity of trigger T1 is finished in EDD Entity E1. Now the activity of trigger T2 can be started.

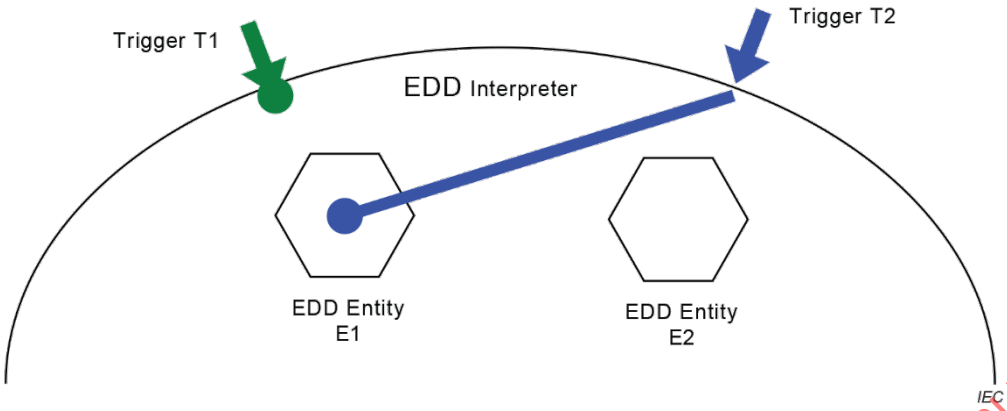

Figure C.5 – Concurrent execution of two triggers (step 4)

C.3 Deadlock detection in concurrent execution

A deadlock situation is shown in Figure C.6. The activity of trigger T1 wants to access EDD Entity E2 as a sub activity of an activity in EDD Entity E1 and trigger T2 wants to access E1 as a sub activity of an activity in E2 at the same time. Both activity chains deadlock one another.

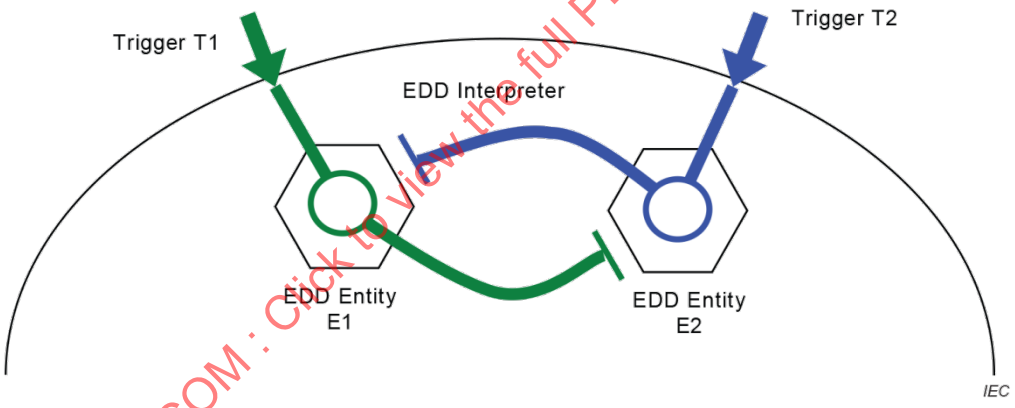

Figure C.6 – Concurrent execution of two triggers

In fact, the activity chain of trigger T1 is blocked by the activity chain of trigger T2 and vice versa. The above example is one of the simplest deadlock scenarios, probably it will happen that much more complex deadlock scenarios occur that usually have a couple of more activity chains involved.

Independently from the complexity of a deadlock scenario, there exists a simple rule to detect deadlocks by monitoring blocking dependencies of activity chains. If activity chain of trigger T(n) is blocked by activity chain of trigger T(n + 1), and T(n + 1) is blocked by T(n + 2), a deadlock scenario is reached when this relation circles back and a trigger T(m) is blocked by T(n). It is not a deadlock scenario as long as the series of blocking dependencies ends up in a non-blocked activity chain.

Usually, it can be expected that the reason for a deadlock scenario is found in an involved device package. Therefore, device package developers should use cross-block and cross-module references only with care and caution.

Nevertheless, the FDI Server is responsible for detecting deadlock scenarios. If the FDI Server has detected a deadlock scenario, it can break it by aborting one of the involved activity chains and even recall the aborted trigger at a later time.

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

INTÉGRATION DES APPAREILS DE TERRAIN (FDI) –

Partie 3: Serveur

AVANT-PROPOS

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

La Norme internationale IEC 62769-3 a été établie par le sous-comité 65E: Les dispositifs et leur intégration dans les systèmes de l'entreprise, du comité d'études 65 de l'IEC: Mesure, commande et automation dans les processus industriels.

Cette deuxième édition annule et remplace la première édition parue en 2015. Cette édition constitue une révision technique.

Cette édition inclut les modifications techniques majeures suivantes par rapport à l'édition précédente:

a) modification du concept de contexte rédactionnel en vue de l'harmonisation des séries IEC 61804 et IEC 62769.

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

FDIS	Rapport de vote
65E/760/FDIS	65E/770/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 internationale.

La version française de la norme n'a pas été soumise au vote.

Ce document a été rédigé 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 ce document 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 au document recherché. À cette date, le document sera

- reconduit,
- supprimé,
- remplacé par une édition révisée, ou
- amendé.

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 série IEC 62769 est publiée sous le titre général "*Intégration des appareils de terrain* (*FDI*)" et comporte les parties suivantes:

– Partie 1: Vue d'ensemble
– Partie 2: Client FDI
– Partie 3: Serveur FDI
– Partie 4: Paquetages FDI
– Partie 5: Modèle d'Information FDI
– Partie 6: Mapping de technologies FDI
– Partie 7: Appareils de Communication FDI
– Partie 100: Profils – Extensions de protocoles génériques
– Partie 101-1: Profils – Foundation Fieldbus H1
– Partie 101-2: Profils – Foundation Fieldbus HSE
– Partie 103-1: Profils – PROFIBUS
– Partie 103-4: Profils – PROFINET
– Partie 109-1: Profils – HART et WirelessHART
– Partie 115-2: Profils – Définitions spécifiques au protocole pour Modbus-RTU
– Partie 150-1: Profils – ISA 100.11a

INTÉGRATION DES APPAREILS DE TERRAIN (FDI) –

Partie 3: Serveur

1 Domaine d'application

La présente partie de l'IEC 62769 spécifie le Serveur FDI. L'architecture FDI complète est représentée à la Figure 1. Les composants architecturaux qui relèvent du domaine d'application du présent document ont été mis en évidence dans cette figure.

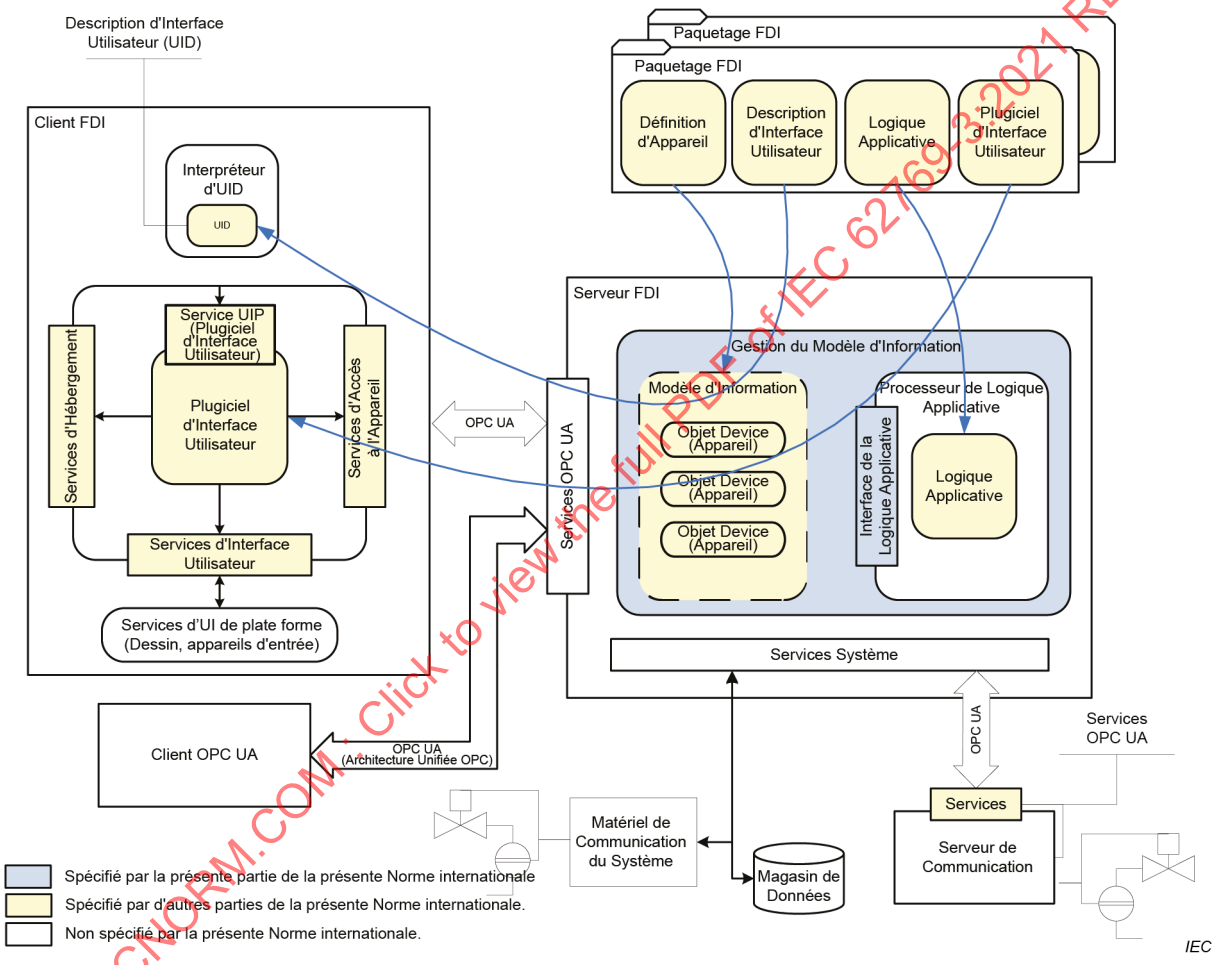

Figure 1 – Diagramme de l'architecture FDI

2 Références normatives

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

IEC 61804 (toutes les parties), *Blocs fonctionnels (FB) pour les procédés industriels et le langage de description électronique de produit (EDDL)*

IEC 61804-4 :2020, *Blocs fonctionnels (FB) pour les procédés industriels et le langage de description électronique de produit (EDDL) – Partie 4: Interprétation EDD*

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

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

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

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

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

IEC 62769-5, *Intégration des appareils de terrain (FDI) – Partie 5: Modèle d'Information FDI*

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

3 Termes, définitions, termes abrégés et conventions

3.1 Termes et définitions

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

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

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

3.1.1
proxy d'actions
entité interne du Serveur FDI qui encapsule toutes les méthodes EDD spécifiées dans une définition d'Action EDD

Note 1 à l'article: L'abréviation "FDI" est dérivée du terme anglais développé correspondant "*Field Device Integration*".

Note 2 à l'article: L'abréviation "EDD" est dérivée du terme anglais développé correspondant "*Electronic Device Definition*".

3.2 Termes abrégés

Pour les besoins du présent document, les termes abrégés de l'IEC 62769-1 s'appliquent.

3.3 Conventions

Pour les besoins du présent document, les conventions de l'IEC 62769-1 s'appliquent.

4 Vue d'ensemble

La structure d'un Serveur FDI est représentée à la Figure 1.

Les Serveurs de FDI qui prennent en charge la connectivité avec des Clients FDI tiers doivent prendre en charge l'OPC UA. Un fournisseur peut fournir aussi bien un Serveur FDI qu'un ou plusieurs Clients FDI. Dans ce cas, les Clients FDI peuvent communiquer avec le Serveur FDI par des protocoles propriétaires.

Un Serveur FDI communique avec des appareils par une Communication native (voir 7.2.1) et/ou par l'intermédiaire d'Appareils de Communication (voir l'IEC 62769-7).

Un Serveur FDI fournit des informations aux Clients FDI par l'intermédiaire d'un Modèle d'Information (voir l'IEC 62769-5) comme suit.

- Le Modèle d'information inclut des informations relatives aux Types d'Appareils et aux Instances d'Appareils. Les informations relatives à une Instance d'Appareil comprennent des données hors ligne (données d'ingénierie) ainsi que des données en ligne (valeurs issues de l'appareil physique).
- Le Modèle d'Information est créé en utilisant des informations issues de Paquetages FDI. Toutefois, toutes les informations d'un Paquetage FDI ne sont pas reflétées dans le Modèle d'Information.

- L'intégrité référentielle du Modèle d'Information est maintenue en utilisant des informations issues de Paquetages FDI.
- Les Paquetages FDI peuvent contenir des Attachments (Pièces jointes) qui contiennent des manuels d'appareils et des informations spécifiques à un protocole (voir l'IEC 62769-4). Ces Pièces jointes, y compris les manuels d'appareils et les fichiers de prise en charge spécifiques à un protocole, sont exposées par l'intermédiaire du Modèle d'Information.
- Les Paquetages d'Appareils FDI contiennent des informations relatives aux types d'appareils (voir l'IEC 62769-4). Chaque type d'appareil défini dans un paquetage est mappé sur un nœud DeviceType distinct dans le Modèle d'Information.
- Des Paquetages de Profils FDI sont utilisés pour assurer l'interaction avec les appareils pour lesquels il n'existe pas de Paquetage d'Appareil FDI (voir l'IEC 62769-4).
- Les multiples révisions d'un Paquetage FDI génèrent des nœuds DeviceType distincts dans le Modèle d'Information (voir l'IEC 62769-4).

Les Paquetages FDI contiennent des signatures numériques qui permettent à un Serveur FDI d'authentifier leur contenu (voir l'IEC 62769-4). Un Serveur FDI ne doit pas utiliser un Paquetage FDI si la signature numérique fournie par le Paquetage FDI n'est pas valide.

Un Serveur FDI doit vérifier la Version de Technologie FDI (voir l'IEC 62769-1) de tout Paquetage FDI qui l'utilise pour assurer que ce Paquetage FDI est bien compatible avec le Serveur FDI.

La structure fonctionnelle résultante d'un serveur FDI est décrite à l'Annexe A.

5 Modèle d'Information

5.1 Généralités

Le Serveur FDI doit utiliser la Définition d'Appareil d'un Paquetage FDI pour maintenir le Modèle d'Information.

La Définition d'Appareil peut contenir des expressions conditionnelles. Des expressions conditionnelles sont utilisées lorsqu'un certain aspect de la Définition d'Appareil n'est pas statique, mais dépend plutôt de l'état de l'appareil. Chaque fois que les valeurs en ligne ("online") ou hors ligne ("offline") d'une Instance d'Appareil sont modifiées, le Serveur FDI doit réévaluer les expressions conditionnelles pertinentes et modifier le Modèle d'Information en conséquence.

L'évaluation des expressions conditionnelles peut invalider des variables dans le Modèle d'Information. Le Serveur FDI doit modifier l'attribut AccessLevel des variables invalidées afin qu'elles ne soient plus lisibles ou inscriptibles et le statut de ces variables doit être défini sur "bad" (mauvais). Les demandes de services Read (Lecture) et Write (Écriture) relatives à des variables invalidées ne doivent pas aboutir (échec).

La Définition d'Appareil peut spécifier des relations entre variables dans un appareil. Ces relations peuvent influer sur la valeur des variables dans le Modèle d'Information.

Le Serveur FDI doit générer des Notifications de DataChange (Modification de données) à l'intention de tous les Clients FDI qui s'abonnent à des éléments du Modèle d'Information qui ont varié.

Les Paquetages FDI fournissent une Logique Applicative qui est utilisée par le Serveur FDI pour maintenir l'intégrité du Modèle d'Information. La Logique Applicative spécifiée dans un Paquetage FDI peut invoquer des fonctions builtin qui doivent être mises en œuvre par le Serveur FDI. Les fonctions builtin qui doivent être mises en œuvre par le Serveur FDI sont spécifiées dans l'IEC 61804-5.

5.2 En ligne/Hors ligne

5.2.1 Vue d'ensemble

Le Modèle d'Information maintenu par le Serveur FDI contient des valeurs en ligne et hors ligne. Les valeurs en ligne reflètent les valeurs dans l'appareil/composant physique. Les valeurs hors ligne reflètent les valeurs stockées dans une base de données de configuration.

Les valeurs hors ligne sont mises à jour par l'intermédiaire de demandes de service d'écriture issues d'un Client FDI ou d'une Logique Applicative exécutée par le Serveur FDI. Les valeurs hors ligne ne sont pas mises à jour lorsque le Serveur FDI lit des données provenant de l'appareil ou écrit des données dans l'appareil.

Les valeurs en ligne dans le Modèle d'Information ne sont pas mises à jour par l'intermédiaire des demandes de service d'écriture. Les demandes de service d'écriture abouties dans le Modèle d'Information conduisent à des modifications des valeurs dans les appareils physiques. Les valeurs en ligne dans le Modèle d'Information sont ensuite mises à jour en réponse aux demandes de service de lecture ou d'abonnements.

Les Serveurs FDI peuvent fournir un mécanisme spécifique à un serveur pour créer des Instances d'Appareils sans la présence de matériel physique. Le Serveur FDI crée ces instances en utilisant des informations dans le Paquetage FDI. Toutes les demandes de lecture/écriture pour des valeurs en ligne dans le cas d'Instances d'Appareils sans appareil physique doivent retourner une erreur.

Le transfert d'informations entre les valeurs hors ligne et l'appareil physique est pris en charge par l'intermédiaire des méthodes TransferToDevice et TransferFromDevice dans le Modèle d'Information. Ces Méthodes doivent respectivement mettre en œuvre les procédures de téléchargement descendant et téléchargement montant, telles que spécifiées dans l'IEC 61804-4. Lorsqu'aucune mise en œuvre n'est assurée selon l'IEC 61804-4, ces Méthodes doivent retourner Bad_NotSupported, conformément à l'IEC 62541-4.

L'Appareil doit avoir été verrouillé avant d'invoquer ces méthodes comme cela est spécifié dans l'IEC 62769-5.

5.2.2 Transfer to device (Transfert vers l'appareil)

La méthode TransferToDevice doit mettre en œuvre la procédure de téléchargement descendant telle que spécifiée dans l'IEC 61804-4. Cette procédure permet de transférer les valeurs hors ligne vers l'appareil physique.

En règle générale, il convient que le Serveur FDI ne modifie pas le nœud Variable Online lors de l'écriture d'une valeur dans l'appareil. Il convient que le nœud Variable Online soit mis à jour seulement dans le processus d'opérations d'écriture ou d'abonnements. Néanmoins, comme spécifié dans l'IEC 62769-5, le Serveur FDI réinitialise toutes les éventuelles Valeurs placées en cache pour les Nœuds cibles dans le Modèle d'Information de façon à pouvoir les relire à la demande suivante.

Les informations de statut retournées pour chaque variable incluse dans les demandes de service d'écriture sont utilisées pour composer le TransferResult, comme cela est spécifié dans l'IEC 62769-5.

5.2.3 Transfer from device (Transfert à partir de l'appareil)

La méthode TransferFromDevice doit mettre en œuvre la procédure de téléchargement montant telle que spécifiée dans l'IEC 61804-4. Cette procédure permet de transférer les valeurs de l'appareil physique vers les valeurs hors ligne.

Si l'une quelconque des opérations de lecture dans l'appareil échoue pendant le téléchargement montant, la valeur hors ligne correspondante ne doit pas être modifiée.

Les informations de statut retournées pour chaque variable incluse dans la demande de service de lecture sont utilisées pour composer le TransferResult, comme cela est spécifié dans l'IEC 62769-5.

5.3 Privilèges d'accès

Les systèmes mettent en œuvre des politiques de sécurité et d'accès fondées sur un certain nombre de caractéristiques telles que le rôle de l'utilisateur et la zone dans l'installation. Les Serveurs FDI utilisent ces politiques, conjointement avec les informations dans les Paquetages FDI, pour déterminer les privilèges d'accès accordés à l'utilisateur.

Les éléments d'un Paquetage FDI peuvent être associés à un ou plusieurs attributs d'utilisation. Le Serveur FDI utilise ces attributs pour définir l'attribut UserAccessLevel des Variables et l'attribut UserExecutable des Méthodes. Les attributs d'utilisation dans un Paquetage FDI sont simplement des conseils à utiliser par le Serveur FDI, c'est-à-dire qu'ils peuvent être rejetés ou ignorés par le Serveur FDI. Voir également l'Annexe B.

5.4 Paramètres privés

Les Paramètres et les Actions spécifiés dans un Paquetage FDI peuvent être déclarés comme étant privés. Les Paramètres et Actions privés ne doivent pas être consultables; ils ne doivent être accessibles qu'au moyen de références partant d'autres éléments d'un Paquetage FDI.

Plus spécifiquement, le Serveur FDI doit prendre en charge les Paramètres et Actions privés comme suit.

- Le Serveur FDI doit créer des nœuds dans le Modèle d'Information pour les Paramètres et Actions privés.
- Le Serveur FDI ne doit pas inclure d'informations relatives à des Paramètres et Actions privés dans une réponse à une demande de service Browse (Consulter), BrowseNext (Consulter le suivant), QueryFirst (Interroger le premier) ou QueryNext (Interroger le suivant).
- Le Serveur FDI doit retourner les NodeIds des Paramètres et Actions privés lorsque le nom d'un Paramètre ou d'une Action privé(e) est transmis à TranslateBrowsePathsToNodeIds.
- Le Serveur FDI doit traiter une demande de service de lecture/écriture pour un Paramètre privé de la même manière qu'il le fait pour des Paramètres publics (consultables) (voir 5.7 et 5.8).
- Le Serveur FDI doit exécuter les Actions privées de la même manière qu'il le fait pour les Actions publiques (consultables) (voir 5.12).

Par exemple, des paramètres privés sont des paramètres qu'il convient de ne modifier que par une Action. Il convient que ces paramètres ne soient pas visibles des Clients FDI afin d'empêcher un accès direct. Les Clients FDI invoquent des Actions pour accéder à ces paramètres privés.

5.5 Verrouillage (Locking)

Le Serveur FDI fournit des services de verrouillage pour accorder à des Clients FDI l'accès exclusif à des éléments Appareils et Réseaux dans le Modèle d'information. Les services de verrouillage sont constitués d'un jeu de Méthodes et d'informations de statut. Les Méthodes (et leur comportement) sont spécifiées dans l'IEC 62769-5.

Le comportement ci-après doit être mis en œuvre par le Serveur FDI pour prendre en charge des verrous.

- Le verrouillage s'applique aux nœuds en ligne et hors ligne.
- Après mise en place du verrou par un Client FDI, toute tentative d'écriture d'un Paramètre ou d'exécution d'une Action par un autre Client FDI doit être rejetée.
- Le verrouillage n'est pas exigé pour les services de lecture.
- Les Paramètres qui sont verrouillés par un Client FDI peuvent toujours être lus par d'autres Clients FDI, c'est-à-dire que les demandes de lecture sur un Paramètre qui est verrouillé ne sont pas rejetées.

L'utilisation interne du mécanisme de verrouillage pour maintenir l'intégrité du Modèle d'Information est spécifique à un fournisseur de Serveur FDI.

La Figure 2 représente une séquence de verrouillage avec plusieurs invocations de service pendant l'état verrouillé.

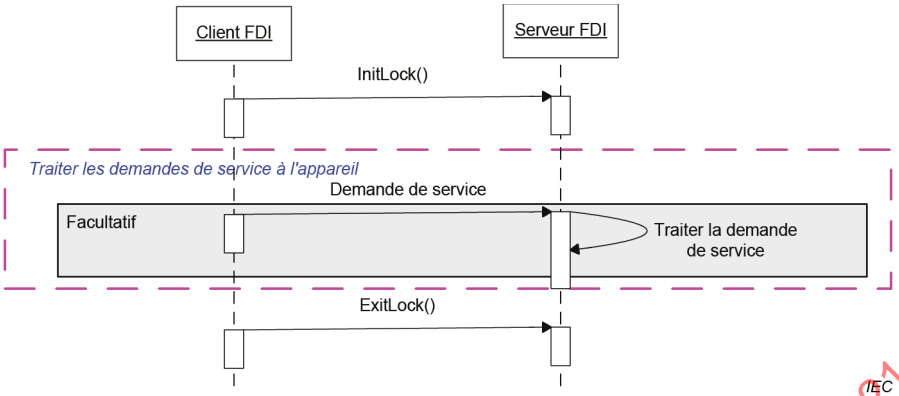

Figure 2 – Services de verrouillage

Une demande de service qui exige le verrouillage doit échouer en partie ou en totalité si aucun verrou n'a été acquis par le Client FDI au moyen d'InitLock avant la demande du service. Le Client FDI doit libérer le verrou au moyen d'ExitLock après l'exécution de toutes les demandes de service.

Les opérations d'écriture échouent en partie, c'est-à-dire qu'elles retournent un code de statut pour chaque variable qui appartient au jeu de variables à écrire, car certaines d'entre elles peuvent appartenir à des appareils qui sont verrouillés et d'autres à des appareils qui ne sont pas verrouillés.

Les Serveurs FDI peuvent placer en file d'attente des demandes InitLock jusqu'à ce qu'un service pour lequel un verrou a été créé s'achève et que le verrou ait été libéré. Cependant, une telle optimisation ne fait pas partie du comportement normalisé exigé d'un Serveur FDI.

5.6 EditContext

5.6.1 Concept et modèle d'utilisation

Le Serveur FDI fournit le modèle EditContext pour interagir avec les Clients au cours de leur tâche de modification. Le concept est étroitement lié aux UID et répond aux besoins des dialogues UI pilotés par un Serveur sur la base des règles de l'EDDL.

Un EditContext peut être utilisé pour apporter des modifications à des Valeurs de Variables visibles au Serveur sans les appliquer à la représentation en ligne ou hors ligne d'un Appareil. Le Serveur applique la logique applicative associée à la Variable modifiée qui – dans certains cas – induit des modifications à d'autres Valeurs de Variables (par exemple, si une unité technique est modifiée) ou à l'UID (par exemple, une Variable devient invisible). Ainsi, le Client peut utiliser un EditContext pour modifier ("éditer") des Paramètres tels que des unités techniques, des plages et plus, vérifier les éventuels effets secondaires, et réajuster les valeurs de réglage avant d'appliquer les modifications.

Un Serveur FDI peut mettre en œuvre différentes stratégies d'EditContext:

- Une seule instance d'EditContext pour tous les dialogues d'un Client FDI.
- Plusieurs instances d'EditContext.
- Des instances hiérarchiques d'EditContext.

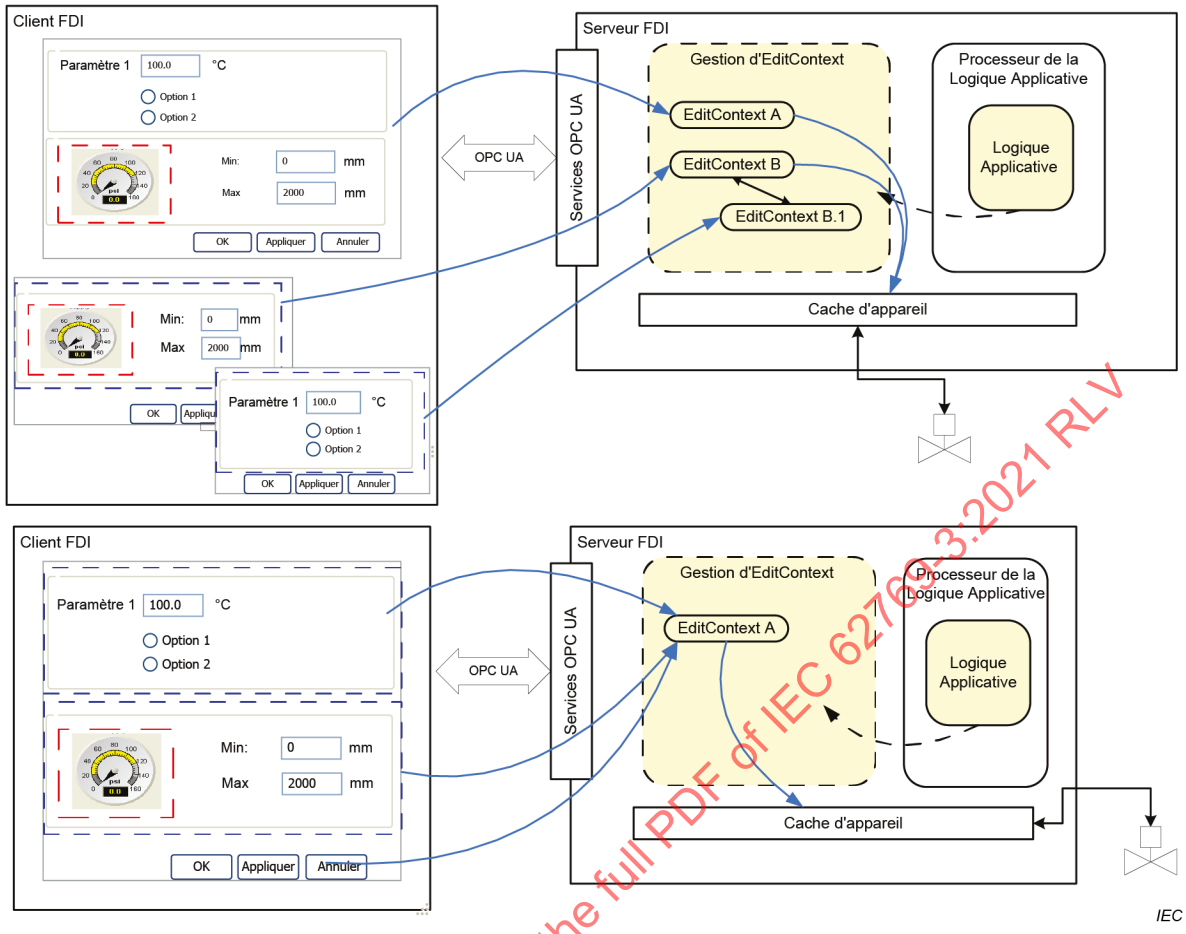

Figure 3 – Modèles d'EditContext

La Figure 3 représente deux stratégies possibles du Serveur, ainsi que le mode d'adaptation du Client. Dans le scénario du bas, le Serveur fournit une seule instance d'EditContext pour tous les dialogues. Dans le cas présent, le Client regroupe tous les dialogues et expose un seul jeu de boutons pour Apply (Appliquer) et Cancel (Annuler), car il concerne toujours toutes les modifications.

Dans le scénario du haut, le Serveur fournit plusieurs instances d'EditContext, l'une étant l'enfant d'une autre. Chaque instance peut être adressée séparément. Lorsque les modifications dans une instance enfant sont appliquées, elles sont transférées au parent. Lorsque les modifications dans une instance racine sont appliquées, elles sont transférées à l'Appareil.

Les dépendances parent-enfant sont définies à l'Article 8 de l'IEC 61804-4:2020.

5.6.2 Services

Un jeu de Services est fourni au Client FDI pour maintenir des instances d'EditContext (voir l'IEC 62769-5 pour une description détaillée de ces Services):

- GetContext – Ce Service est utilisé pour demander une instance d'EditContext. Le Client spécifie certaines caractéristiques pour que le Serveur décide de l'instance d'EditContext à retourner. Selon sa stratégie interne, le Serveur retourne la même instance ou de nouvelles instances.
- RegisterNodes – Le Client FDI doit enregistrer tous les Nodes (Nœuds) du Modèle d'Information qui doivent être maintenus dans un EditContext. Il est possible d'enregistrer les Nœuds de la représentation en ligne et hors ligne d'un Appareil. Le résultat est constitué des nouveaux NodeIds que le Client doit utiliser pour appeler des Services pour lire ou écrire des Variables ou s'y abonner ou pour invoquer des Actions.

- Apply – Transférer les Valeurs de Variables modifiées au parent (une instance EditContext parente ou l'Appareil). Si la même Variable a été modifiée dans l'instance parente, elle est écrasée en écriture avec un appel du Service Apply pour l'enfant.
- Reset – Efface toutes les modifications. Une Reset (Réinitialisation) de modifications déjà appliquées n'est pas possible.
- Discard – Supprime une instance d'EditContext (et ses enfants). Les Valeurs modifiées qui n'ont pas été appliquées sont rejetées. Après suppression, tous les NodeId enregistrés sont non valides. Si de tels NodeId font encore l'objet d'abonnement, le Client reçoit une notification avec les StatusCode corrects.

Le Client appelle tout d'abord GetEditContext pour acquérir une instance d'EditContext. Il enregistre ensuite les Nœuds qu'il souhaite y insérer. L'enregistrement retourne les nouveaux NodeId qui peuvent alors être utilisés pour lire ou écrire des Variables ou s'y abonner et pour appeler des Méthodes.

Le Client peut appeler GetEditContext plusieurs fois, par exemple lorsqu'il ouvre une fenêtre d'édition supplémentaire, ou pour un dialogue totalement distinct (diagnostic en parallèle à la configuration). Il relève de la stratégie du Serveur de décider de retourner une nouvelle instance ou la même instance. Le Client est réputé adapter son interface utilisateur à la stratégie d'EditContext du Serveur. Voir à la Figure 3 comment les Clients peuvent positionner leurs boutons Apply et Cancel afin que l'Utilisateur comprenne clairement quelles modifications sont appliquées ou rejetées.

5.6.3 NodeId

RegisterNode retourne deux NodeId pour chaque Nœud enregistré: un ContextNodeId et un DeviceNodeId. Le Client utilise ces NodeId pour appeler des Services OPC UA afin de lire, écrire ou s'abonner ou pour appeler une Méthode.

L'utilisation de ContextNodeId adresse la Valeur dans l'instance d'EditContext. L'utilisation de DeviceNodeId adresse la Valeur dans l'Appareil.

5.6.4 Lecture

La lecture d'une Variable ou l'abonnement à une Variable avec ContextNodeId retourne la Valeur modifiée provenant de l'instance d'EditContext. Si aucune Valeur modifiée n'existe, la Valeur provenant de l'instance parente ou de l'Appareil (en ligne ou hors ligne) est retournée.

Le StatusCode indique si la Valeur provient de l'Appareil (StatusCode Good défini dans l'IEC 62541) ou d'une instance d'EditContext (StatusCode Good_Edited défini dans l'IEC 62769-5).

La lecture d'une Variable ou l'abonnement à une Variable avec DeviceNodeId retourne la Valeur issue de l'Appareil (en ligne ou hors ligne).

5.6.5 Écriture

L'écriture dans une Variable avec ContextNodeId modifie la Valeur dans l'instance d'EditContext.

L'écriture dans une Variable avec DeviceNodeId modifie la Valeur dans l'Appareil (en ligne ou hors ligne). Toute Valeur modifiée pour cette Variable dans l'instance d'EditContext adressée ou ses parents est réinitialisée.

5.6.6 Écriture de Variables dominantes et dépendantes

Dans certains cas, la valeur d'une Variable dépend de la valeur d'une autre Variable. La manière dont ces dépendances sont évaluées est spécifiée dans l'IEC 61804-4.

Lorsque de telles Variables sont éditées, le Serveur FDI doit suivre les diagrammes d'états spécifiés dans la Figure 4 pour "Online" (En ligne) et dans la Figure 5 pour "Offline" (Hors ligne). Ces diagrammes spécifient les états et les transitions au cours du processus d'édition de ce type de Variables. Le statut est le StatusCode que les Clients FDI reçoivent avec la Valeur lorsqu'ils surveillent ou lisent ces Variables avec ContextNodeId. Pour les Variables dépendantes, tout StatusCode Good (Correct) ou Uncertain (Incertain) devient

Uncertain_DominantValueChanged et tout StatusCode Bad devient Bad_DominantValueChanged. Pour les Variables dominantes, les StatusCode Good deviennent Good_DependentValueChanged, les StatusCode Uncertain deviennent Uncertain_DependentValueChanged et les StatusCode Bad deviennent Bad_DependentValueChanged.

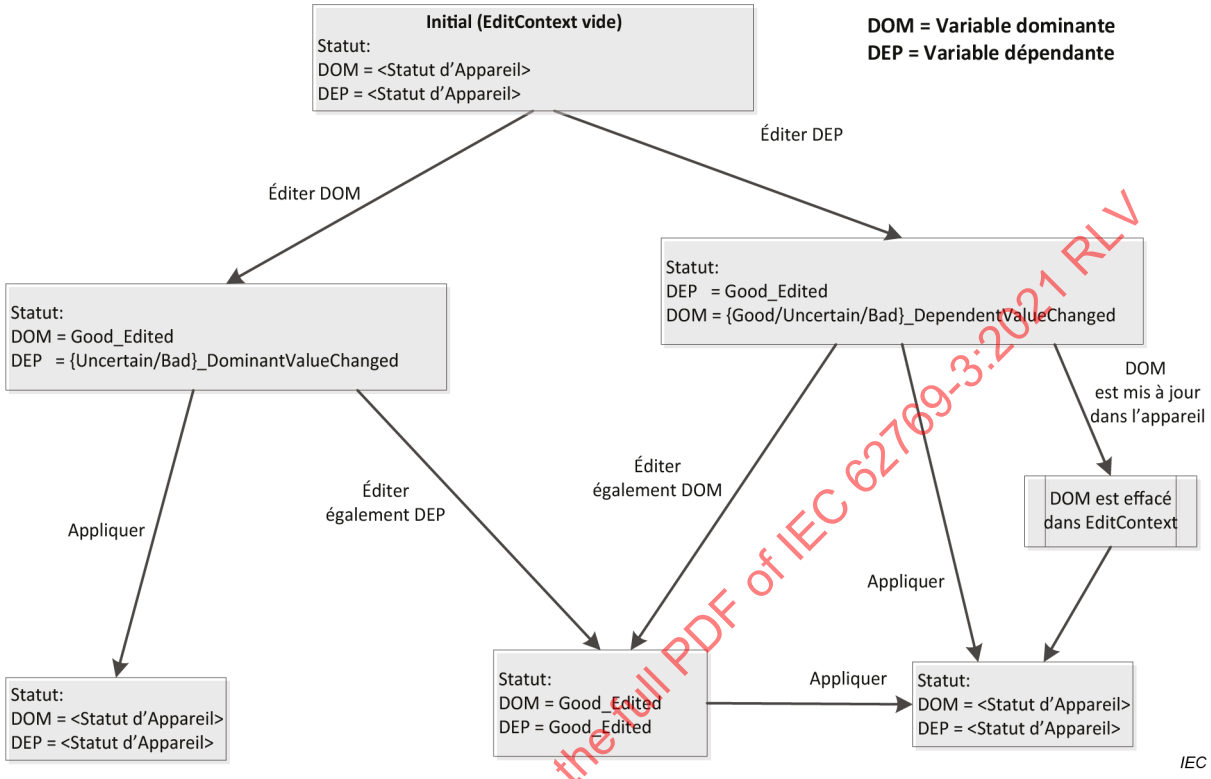

Figure 4 – Diagramme d'états EditContext Online pour Variables dominantes et dépendantes

Si les Variables dominantes et dépendantes doivent être modifiées, il est fortement recommandé, dans le cas de modifications en ligne, d'effectuer ces modifications en sessions successives de modification. Les systèmes peuvent appliquer ce type de sessions.

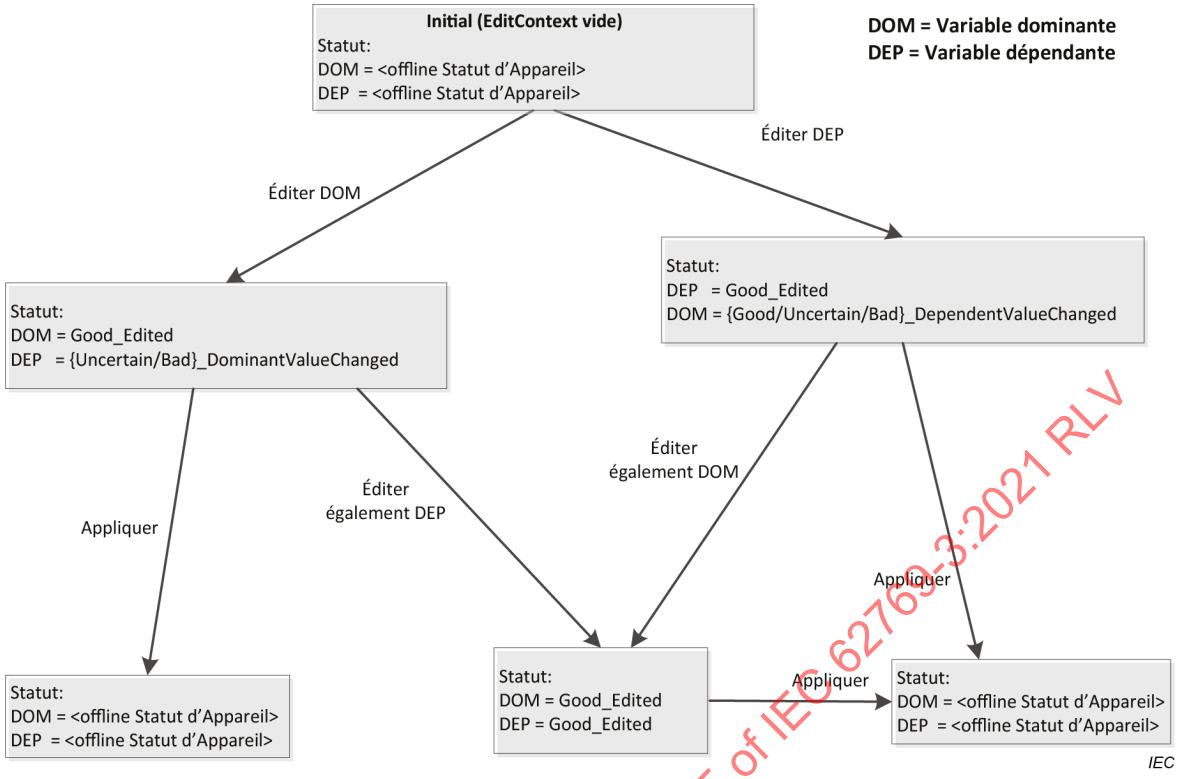

Figure 5 – Diagramme d'états EditContext Offline pour Variables dominantes et dépendantes

5.6.7 Actions (MÉTHODES EDD)

Avant d'invoquer des Actions, le Client doit enregistrer le Nœud ActionSet de l'Appareil. Le NodeId de ce Nœud doit être spécifié lors de l'appel d'InvokeAction.

L'appel d'InvokeAction avec le ContextNodeId du Nœud ActionSet l'associe à l'instance correcte d'EditContext. Le Serveur crée implicitement une instance d'EditContext pour l'Action invoquée. Ceci est représenté à la Figure 6.

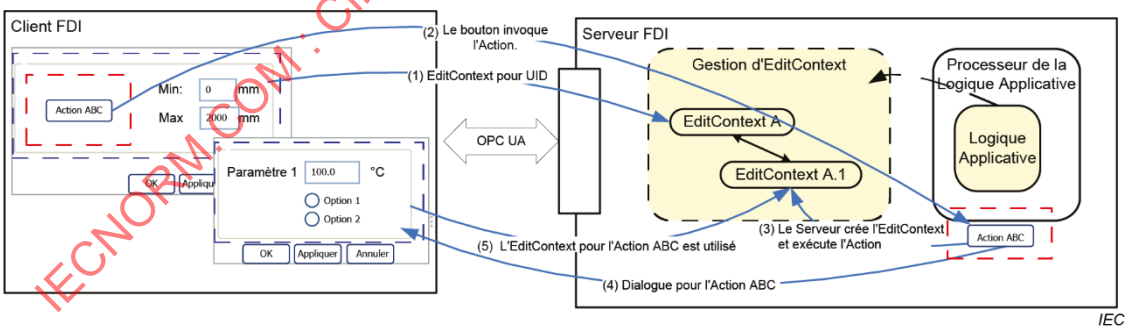

Figure 6 – EditContext pour les Méthodes EDD

La MÉTHODE EDD représentée par l'Action utilise des builtins pour modifier les Valeurs dans l'EditContext ou l'Appareil, pour synchroniser les modifications avec le cache sous-jacent ou pour les rejeter.

Si l'exécution de l'Action échoue, l'EditContext pour l'Action est rejeté.

5.6.8 UID

L'interpréteur d'UID dans le Client appelle GetEditContext avant d'appeler un UID de niveau supérieur. Des EditContext supplémentaires pour des dialogues peuvent être instanciés par le Serveur et transmis au Client à l'intérieur de chaque document d'UID. Voir l'IEC 62769-2 pour le Schéma des UID et le traitement d'un EditContext dans l'Interpréteur d'UID.

5.6.9 Synchronisation

Un Lock (verrou) doit être créé avant que la première Valeur ne soit écrite dans un EditContext.

Le verrouillage est également exigé pour écrire directement dans l'Appareil.

5.7 Lecture

5.7.1 Généralités

Le service Read spécifié dans l'IEC 62541-4 peut être utilisé pour lire une seule valeur ou plusieurs valeurs dans un seul appareil ou dans plusieurs appareils. Si une demande de service Read spécifie que plusieurs valeurs doivent être lues, les valeurs sont lues dans l'ordre de leur apparition dans la demande de service.

Toutes les valeurs qui sont retournées au Client FDI consécutivement à une demande de service Read ne doivent pas être échelonnées.

Un échec qui se produit au cours de la lecture d'une seule valeur ne doit pas abandonner le processus de lecture; toutes les valeurs doivent être lues. Chaque valeur retournée contient un statut indiquant le succès ou l'échec. Les informations de statut normalisées des services OPC UA sont retournées par le Serveur FDI en réponse aux appels de service tels que spécifiés en 6.2.

Un Paquetage FDI peut définir des actions de lecture qui sont exécutées par le Serveur FDI au cours des demandes de service de lecture. Ces actions ne doivent pas exiger d'interaction de l'utilisateur; elles sont strictement destinées à être utilisées pour le traitement de Logique Applicative. Une action de lecture qui exige finalement une interaction de l'utilisateur ne procède pas à l'intégration (builtin), mais retourne une erreur si possible. Les actions de lecture suivantes peuvent être définies dans un Paquetage FDI:

- Actions avant la lecture
- Actions après la lecture

Le Serveur FDI invoque des actions avant la lecture et après la lecture au cours du traitement des demandes de service de lecture relatives aux valeurs en ligne seulement. Ces actions ne sont pas invoquées pendant la lecture de valeurs hors ligne.

Outre les actions de lecture, un Paquetage FDI peut définir des actions de rafraîchissement qui sont exécutées par le Serveur FDI au cours des demandes de service de lecture. Ces actions ne doivent pas exiger d'interaction de l'utilisateur; elles sont strictement destinées à être utilisées pour le traitement de Logique Applicative. Une action de rafraîchissement qui exige finalement une interaction de l'utilisateur ne procède pas à l'intégration (builtin), mais retourne une erreur si possible.

Le Serveur FDI invoque des actions de rafraîchissement au cours du traitement des demandes de service de lecture relatives aux valeurs tant hors ligne qu'en ligne.

Les actions de rafraîchissement mentionnées en 5.7.1 ne doivent pas être confondues avec les relations de rafraîchissement.

NOTE Les actions de rafraîchissement sont définies au moyen de la construction REFRESH_ACTIONS de l'EDDL à l'intérieur d'une construction VARIABLE de l'EDDL. En revanche, les relations de rafraîchissement sont définies au moyen de la construction REFRESH de l'EDDL. Le traitement des relations de rafraîchissement est inclus dans l'événement générique "Process Conditionals/Relations" (Traiter des Conditionnelles/Relations) qui apparaît dans les diagrammes de séquences pour les services read (lecture), write (écriture) et subscription (abonnement). Les explications qui suivent les diagrammes comprennent des relations de rafraîchissement dans le terme général "EDDL relations" (Relations de l'EDDL). Voir l'IEC 61804-3 our de plus amples informations relatives aux actions de rafraîchissement et aux relations de rafraîchissement.

5.7.2 Lecture de variables hors ligne

Le diagramme de séquences à la Figure 7 représente le comportement du Serveur FDI lorsqu'une valeur hors ligne est lue.

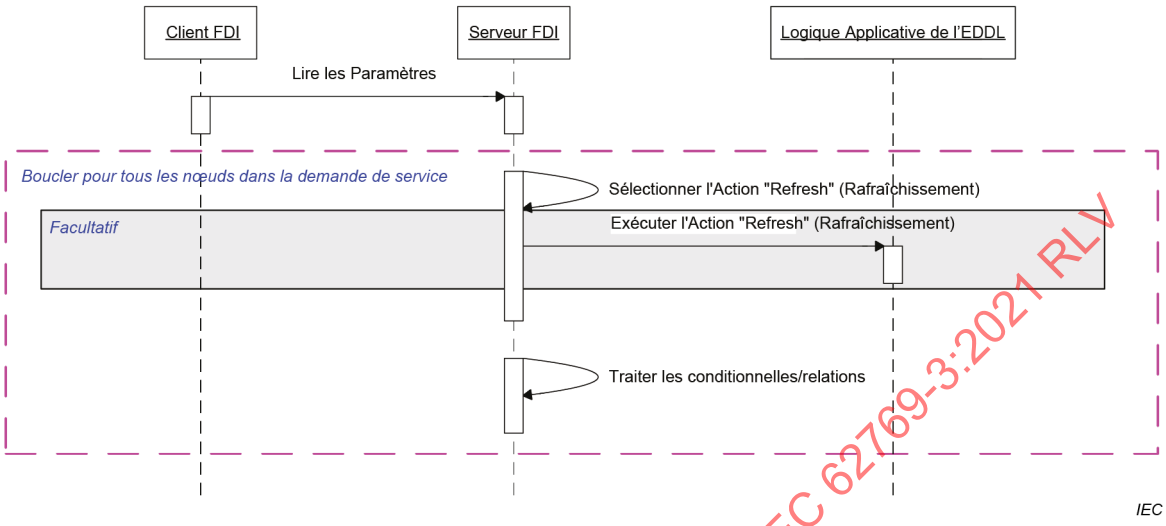

Figure 7 – Variable hors ligne lue

Si une variable a des actions de rafraîchissement qui lui sont associées, le Serveur FDI exécute toujours ces actions indépendamment de la valeur MaxAge.

Si les actions de rafraîchissement échouent, le statut retourné pour la variable en question doit indiquer que la lecture a échoué.

Le Serveur FDI évalue les conditionnelles et les relations après l'exécution des actions de rafraîchissement. Cette évaluation permet de réaliser la réévaluation d'expressions conditionnelles dans la Logique Applicative de l'EDDL et le traitement des relations EDDL.

5.7.3 Lecture de variables en ligne

Le Serveur FDI peut placer en cache les valeurs en ligne lues à partir d'un appareil. Le Serveur FDI maintient un horodatage pour chaque valeur en ligne qui indique le moment auquel la valeur a été lue dans l'appareil. Le Serveur FDI utilise l'argument MaxAge d'une demande de service Read pour déterminer si la valeur placée en cache peut être retournée. Si la différence entre l'horodatage et l'heure actuelle est supérieure à l'argument MaxAge, le Serveur FDI doit lire la valeur dans l'appareil. Autrement, la valeur placée en cache peut être retournée.

Les actions de lecture ne sont exécutées que lorsque la valeur est lue dans l'appareil.

Le diagramme de séquences à la Figure 8 représente le comportement du Serveur FDI lorsqu'une valeur en ligne est lue.

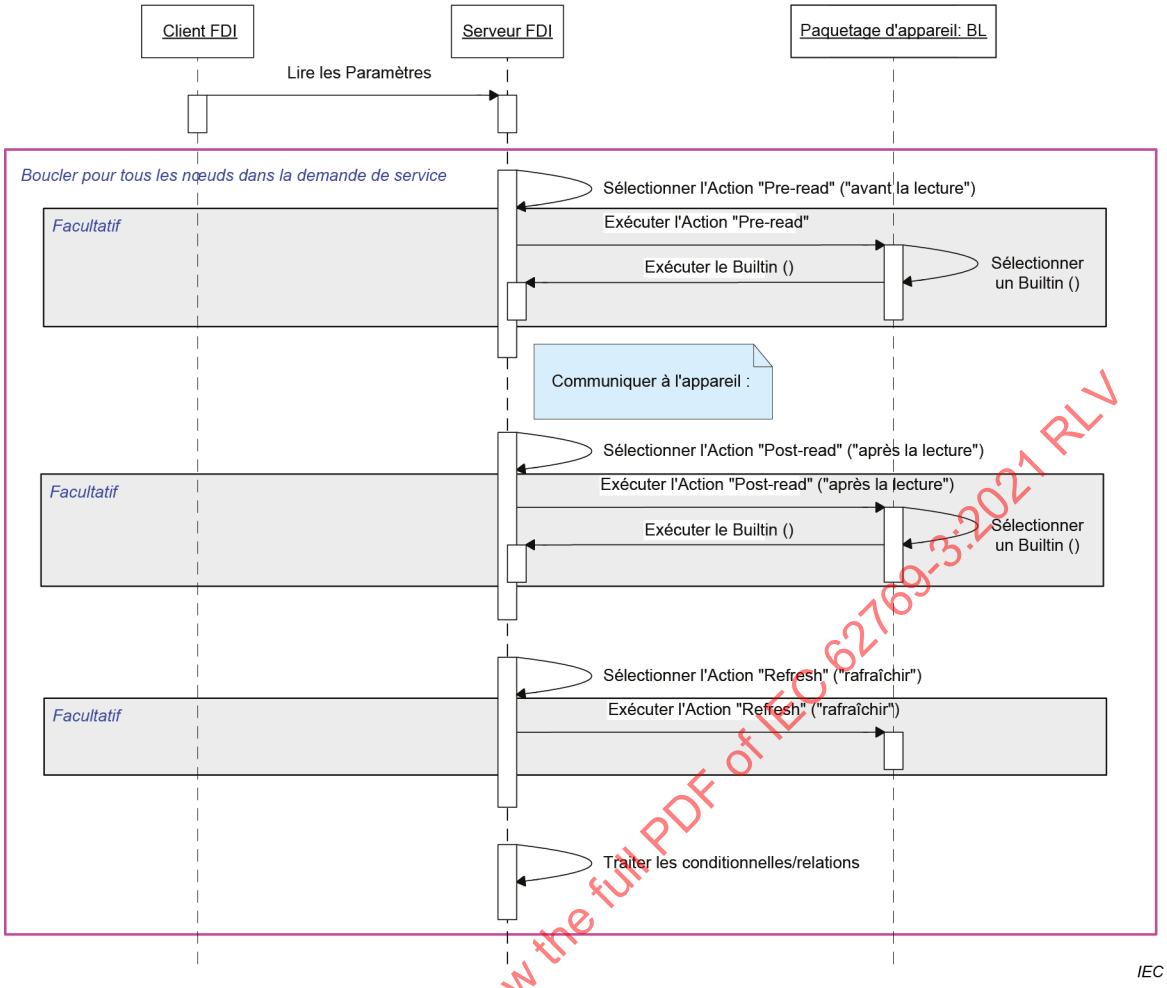

Figure 8 – Variable en ligne lue

Si une variable a des actions avant la lecture qui lui sont associées, ces actions sont exécutées avant la lecture de la variable dans l'appareil. Si une variable a des actions après la lecture qui lui sont associées, ces actions sont exécutées après la lecture de la variable dans l'appareil. Si les actions avant la lecture ou après la lecture échouent, le statut retourné pour la variable en question doit indiquer que la lecture a échoué.

Si une variable a des actions de rafraîchissement qui lui sont associées, ces actions sont traitées comme dans le cas de la lecture d'une variable hors ligne (voir 1.1.1).

Le Serveur FDI évalue les conditionnelles et les relations après que les actions après la lecture ont été exécutées. Cette évaluation permet de réaliser la réévaluation d'expressions conditionnelles dans la Logique Applicative de l'EDDL et le traitement des relations EDDL.

5.8 Écriture

5.8.1 Généralités

Le service Write spécifié dans l'IEC 62541-4 peut être utilisé pour écrire une seule valeur ou plusieurs valeurs dans un seul appareil ou dans plusieurs appareils. Si une demande de service Write spécifie que plusieurs valeurs doivent être écrites, les valeurs sont écrites dans l'ordre de leur apparition dans la demande de service.

Un échec qui se produit au cours de l'écriture d'une valeur unique ne doit pas abandonner le processus d'écriture; toutes les valeurs doivent être écrites. Un statut est retourné indiquant le succès ou l'échec de chaque valeur incluse dans la demande de service. Les informations de statut normalisées des services OPC UA sont retournées par le Serveur FDI en réponse aux appels de service tels que spécifiés en 6.2.

Contrairement à l'opération de lecture, les échecs d'écriture au cours desquels plusieurs variables sont spécifiées peuvent laisser l'appareil dans un état indéfini, avec certaines variables modifiées et d'autres laissées intactes. Il incombe au Client FDI de gérer les échecs partiels.

Il est nécessaire que les Clients FDI verrouillent l'appareil pour l'accès exclusif avant écriture. La demande de verrou peut être émise immédiatement avant la demande de service d'écriture ou elle peut être émise de manière indépendante sur plusieurs demandes de service d'écriture (voir 5.5).

Le Serveur FDI accomplit une validation de données au cours des demandes de service d'écriture relatives aux valeurs en ligne et hors ligne.

Un Paquetage FDI peut définir des actions d'écriture qui sont exécutées par le Serveur FDI au cours des demandes de service d'écriture. Ces actions ne doivent pas exiger d'interaction de l'utilisateur; elles sont strictement destinées à être utilisées pour le traitement de Logique Applicative. Une action d'écriture qui exige finalement une interaction de l'utilisateur ne procède pas à l'intégration (builtin), mais retourne une erreur si possible. Les actions d'écriture suivantes peuvent être définies dans un Paquetage FDI:

- Actions avant l'écriture
- Actions après l'écriture

Le Serveur FDI invoque ces actions au cours du traitement des demandes de service de lecture relatives aux valeurs en ligne seulement. Ces actions ne sont pas invoquées pendant l'écriture de valeurs hors ligne.

5.8.2 Écriture de variables hors ligne

Le diagramme de séquences à la Figure 9 représente le comportement du Serveur FDI lorsqu'une valeur hors ligne est écrite.

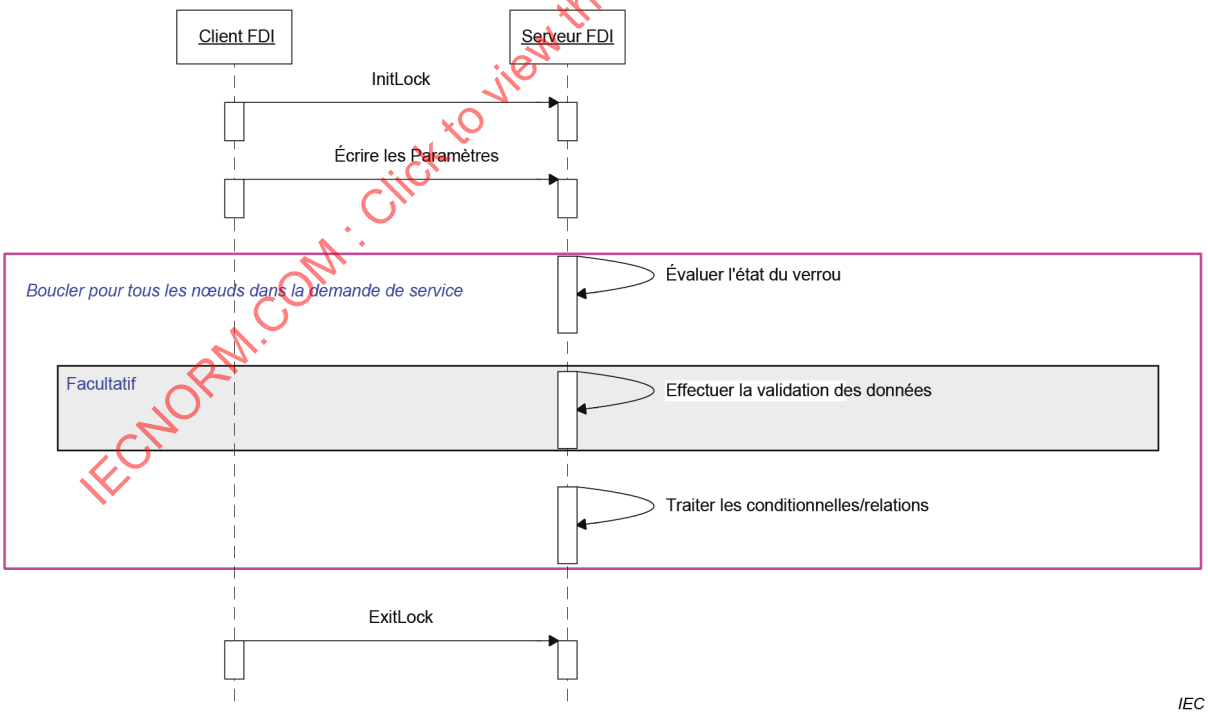

Figure 9 – Écriture immédiate de variables hors ligne

Comme point de départ de l'écriture d'une variable, le Serveur FDI vérifie si l'appareil est verrouillé par le Client FDI. S'il n'est pas verrouillé, le statut retourné pour la variable en question doit indiquer que l'écriture a échoué.

Si l'appareil est verrouillé par le Client FDI, le Serveur FDI effectue la validation des données. La validation consiste fondamentalement en une vérification de la plage et du type en fonction d'informations EDDL. Si la validation de type échoue, le statut retourné pour la variable en question doit indiquer que l'écriture a échoué. Si la validation de plage échoue, le statut retourné pour la variable en question doit indiquer que l'écriture a réussi, mais les informations de statut de la valeur de la variable dans le Modèle d'Information doivent indiquer qu'il est bad (mauvais) ou out-of-range (hors plage).

Si le processus de validation réussit, le Serveur FDI écrit dans la valeur hors ligne de la variable dans le Modèle d'Information.

Après écriture de la valeur de la variable, le Serveur FDI évalue les conditionnelles et les relations. Cette évaluation permet de réaliser la réévaluation d'expressions conditionnelles dans la Logique Applicative de l'EDDL et le traitement des relations EDDL.

5.8.3 Écriture de variables en ligne

Le diagramme de séquences à la Figure 10 représente le comportement du Serveur FDI lorsqu'une valeur en ligne est écrite.

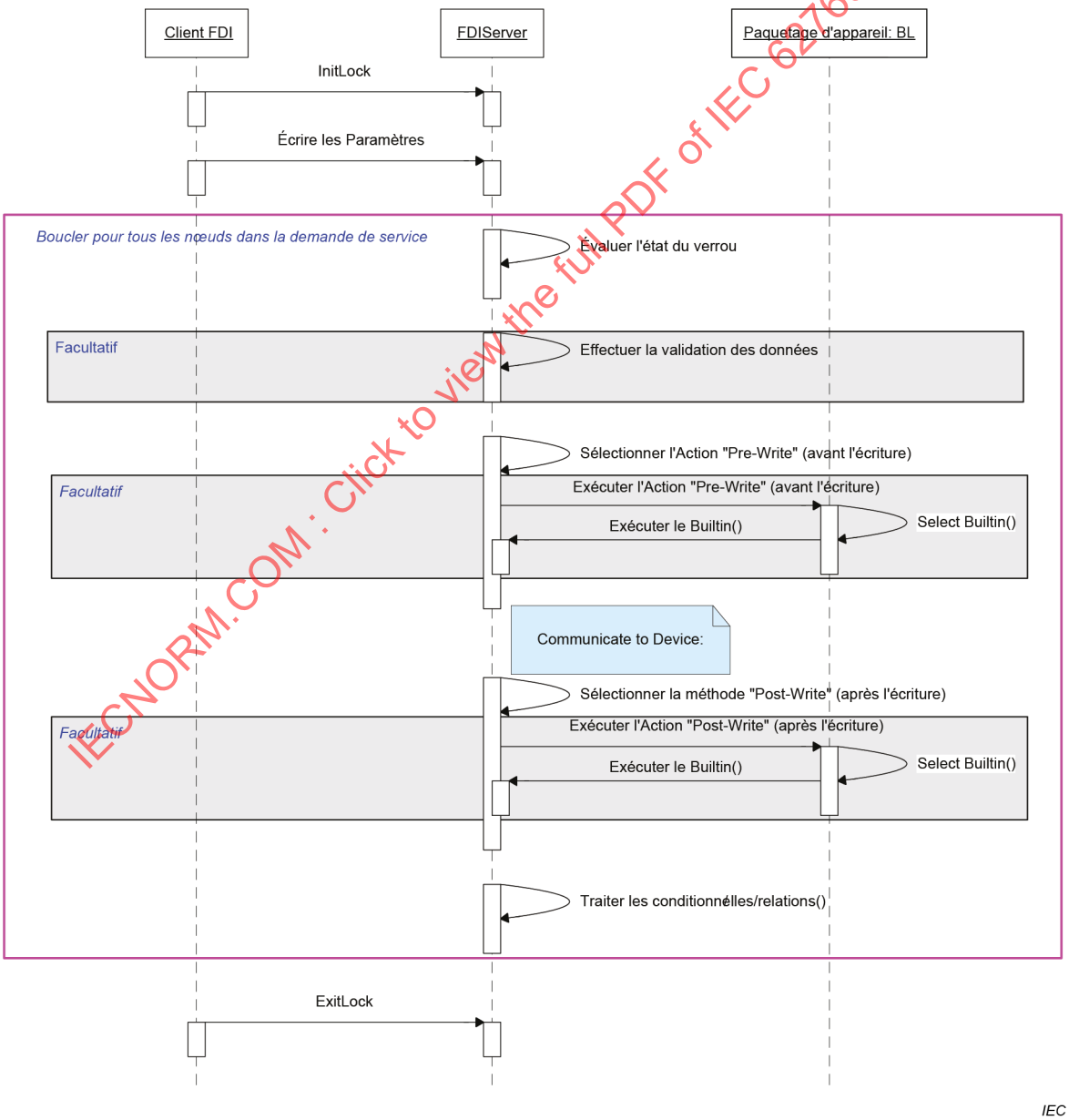

Figure 10 – Écriture immédiate de variable en ligne

Lors de l'écriture d'une variable en ligne, le Serveur FDI vérifie si l'appareil est verrouillé par le Client FDI et accomplit une validation de données telle que décrite en 5.8.2.

Si le processus de validation réussit, le Serveur FDI écrit la variable dans l'appareil physique.

Si une variable a des actions avant l'écriture qui lui sont associées, ces actions sont exécutées avant l'écriture de la variable dans l'appareil. Si les actions avant l'écriture échouent, le statut retourné pour la variable en question doit indiquer que l'écriture a échoué et l'opération d'écriture se termine sans écriture dans l'appareil.

Si une variable a des actions après l'écriture qui lui sont associées, ces actions sont exécutées après l'écriture de la variable dans l'appareil. Si les actions après l'écriture échouent, le statut retourné pour la variable ne doit pas indiquer que l'écriture a échoué, car la valeur a déjà été écrite dans l'appareil. Le statut retourné doit être Good_PostActionFailed.

Le Serveur FDI évalue les conditionnelles et les relations après exécution des actions après l'écriture. Cette évaluation permet de réaliser l'évaluation d'expressions conditionnelles dans la Logique Applicative de l'EDDL et le traitement des relations EDDL.

5.8.4 Écriture dans un EditContext

L'EditContext est spécifié en 5.6.

Le diagramme de séquences à la Figure 11 représente le comportement général d'un EditContext lorsque des Valeurs sont modifiées et appliquées.

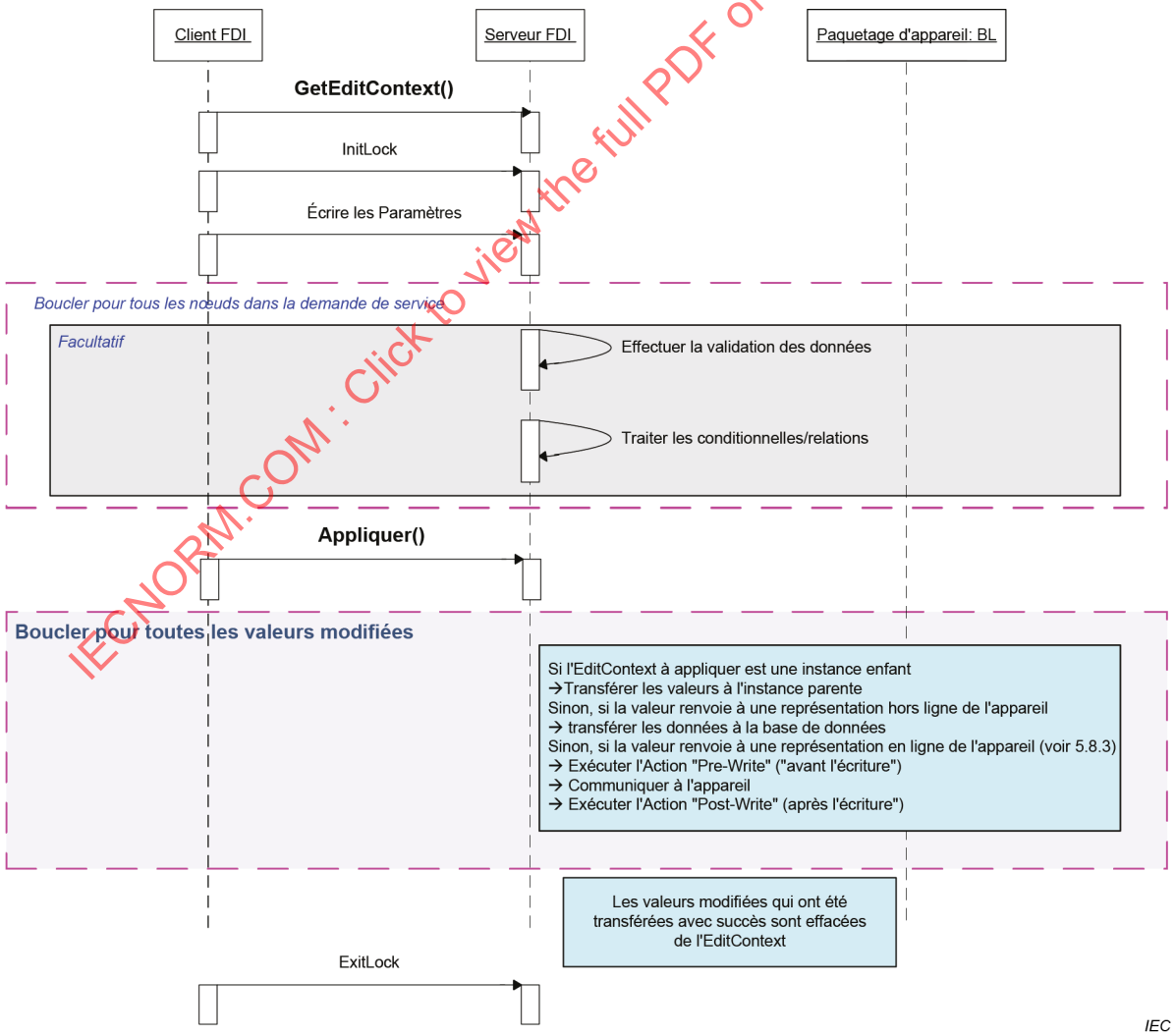

Figure 11 – Écriture avec EditContext

Lors de l'écriture de Variables dans un EditContext, le Serveur FDI effectue la validation des données comme dans le cas des écritures normales dans des données en ligne ou hors ligne. Il traite également les Conditionnelles/Relations. Les modifications apportées aux autres Variables en raison de ce traitement sont également écrites dans l'EditContext.

Lors de l'appel "Apply", les Variables modifiées sont transférées au parent. Si le parent est l'Appareil et si une variable a des actions avant écriture qui lui sont associées, ces actions sont exécutées avant l'écriture de la variable dans l'Appareil. Si les actions avant l'écriture échouent, le statut retourné pour Apply doit indiquer l'erreur.

Si le parent est l'Appareil et si une variable a des actions après l'écriture qui lui sont associées, ces actions sont exécutées après l'écriture de la variable dans l'Appareil. Si les actions après l'écriture échouent, le statut retourné pour la variable ne doit pas indiquer que l'écriture a échoué, car la valeur a déjà été écrite dans l'appareil. Le statut retourné doit être Good_PostActionFailed.

5.9 Subscription (Abonnement)

5.9.1 Généralités

Le service Subscription spécifié dans l'IEC 62541-4 peut être utilisé pour lancer la surveillance d'une seule variable ou de plusieurs variables à partir d'un seul appareil ou de plusieurs appareils.

Un échec relatif à une seule variable lors de l'établissement d'un abonnement à plusieurs variables ne doit pas abandonner le processus d'abonnement; toutes les variables doivent être surveillées. Chaque variable retournée contient un statut indiquant le succès ou l'échec. Les informations de statut normalisées des services OPC UA sont retournées par le Serveur FDI en réponse aux appels de service tels que spécifiés en 6.2.

Un Paquetage FDI peut définir des actions de lecture qui sont exécutées par le Serveur FDI au cours du processus de surveillance d'un abonnement. Ces actions ne doivent pas exiger d'interaction de l'utilisateur; elles sont strictement destinées à être utilisées pour le traitement de Logique Applicative. Une action de lecture qui exige finalement une interaction de l'utilisateur ne procède pas à l'intégration (builtin), mais retourne une erreur si possible. Les actions de lecture suivantes peuvent être définies dans un Paquetage FDI:

- Actions avant la lecture
- Actions après la lecture

Le Serveur FDI invoque ces actions au cours de la surveillance des variables en ligne seulement; ces actions ne sont pas invoquées lors de la surveillance de variables hors ligne.

Outre les actions de lecture, un Paquetage FDI peut définir des actions de rafraîchissement qui sont exécutées par le Serveur FDI au cours du processus de surveillance. Le Serveur FDI invoque ces actions de rafraîchissement au cours de la surveillance des variables tant hors ligne qu'en ligne. Ces actions ne doivent pas exiger d'interaction de l'utilisateur; elles sont strictement destinées à être utilisées pour le traitement de Logique Applicative. Une action de rafraîchissement qui exige finalement une interaction de l'utilisateur ne procède pas à l'intégration (builtin), mais retourne une erreur si possible.

L'intervalle d'échantillonnage demandé par le Client FDI et établi par le Serveur FDI définit un intervalle de temps qui est utilisé pour rechercher périodiquement des modifications apportées à la valeur ou au statut des variables. À chaque intervalle de temps, les actions sont invoquées et la valeur et le statut sont comparés à la valeur et au statut précédents. Une modification de la valeur ou du statut amène le Serveur FDI à préparer une notification de la nouvelle valeur et du nouveau statut.

5.9.2 Abonnement aux variables hors ligne

Le diagramme de séquences à la Figure 12 représente le comportement du Serveur FDI lorsqu'une valeur hors ligne est surveillée.