BS EN 62769-3:2015

BSI Standards Publication

# Devices and integration in enterprise systems; Field Device Integration

Part 3: FDI Server

...making excellence a habit.™

## National foreword

This British Standard is the UK implementation of EN 62769-3:2015. It is identical to IEC 62769-3:2015.

The UK participation in its preparation was entrusted to Technical Committee AMT/7, Industrial communications: process measurement and control, including fieldbus.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

ISBN 978 0 580 78325 8
ICS 25.040.40; 35.100

**Compliance with a British Standard cannot confer immunity from legal obligations.**

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 July 2015.

### Amendments/corrigenda issued since publication

| Date | Text affected |
|---|---|

EUROPEAN STANDARD

NORME EUROPÉENNE

EUROPÄISCHE NORM

**EN 62769-3**

July 2015

ICS 25.040.40; 35.100

English Version

# Devices and integration in enterprise systems; Field Device Integration - Part 3: FDI Server (IEC 62769-3:2015)

Les dispositifs et leur intégration dans les systèmes de l'entreprise; Intégration des appareils de terrain (FDI) - Partie 3: Serveur FDI (IEC 62769-3:2015)

Feldgeräteintegration (FDI) - Teil 3: FDI-Server (IEC 62769-3:2015)

CENELEC

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

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

## European foreword

The text of document 65E/346/CDV, future edition 1 of IEC 62769-3, prepared by SC 65E “Devices and integration in enterprise systems” of IEC/TC 65 “Industrial-process measurement, control and automation" was submitted to the IEC-CENELEC parallel vote and approved by CENELEC as EN 62769-3:2015.

The following dates are fixed:

| | | |
|---|---|---|
| • latest date by which the document has to be implemented at national level by publication of an identical national standard or by endorsement | (dop) | 2016-03-24 |
| • latest date by which the national standards conflicting with the document have to be withdrawn | (dow) | 2018-06-24 |

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

## Endorsement notice

The text of the International Standard IEC 62769-3:2015 was approved by CENELEC as a European Standard without any modification.

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

| | | |
|---|---|---|
| IEC 61804-5 | NOTE | Harmonized as EN 61804-5[1)] |
| IEC 62769-6 | NOTE | Harmonized as EN 62769-6 |

1) To be published.

## Annex ZA
(normative)

## Normative references to international publications with their corresponding European publications

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

NOTE 1 When an International Publication has been modified by common modifications, indicated by (mod), the relevant EN/HD applies.

NOTE 2 Up-to-date information on the latest versions of the European Standards listed in this annex is available here: www.cenelec.eu.

| Publication | Year | Title | EN/HD | Year |
|---|---|---|---|---|
| IEC 61804 | series | Function Blocks (FB) for process control | EN 61804 | series |
| IEC 61804-3 | - | Function blocks (FB) for process control and EDDL - Part 3: EDDL specification and communication profiles | - | - |
| IEC 61804-4 | - | Function blocks (FB) for process control -- Part 4: EDD interpretation | - | - |
| IEC 62541-4 | - | OPC Unified Architecture - Part 4: Services | EN 62541-4 | - |
| IEC 62541-7 | - | OPC unified architecture - Part 7: Profiles | EN 62541-7 | - |
| IEC 62541 | series | OPC unified architecture | EN 62541 | series |
| 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: FDI Communication Devices | - | - |

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

___________

# FIELD DEVICE INTEGRATION (FDI) –

# Part 3: FDI Server

## FOREWORD

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.

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

| CDV | Report on voting |
|---|---|
| 65E/346/CDV | 65E/423/RVC |

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

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

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

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

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

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

## INTRODUCTION

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

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

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

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

a) ABB Research Ltd
   Claes 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.

# 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.

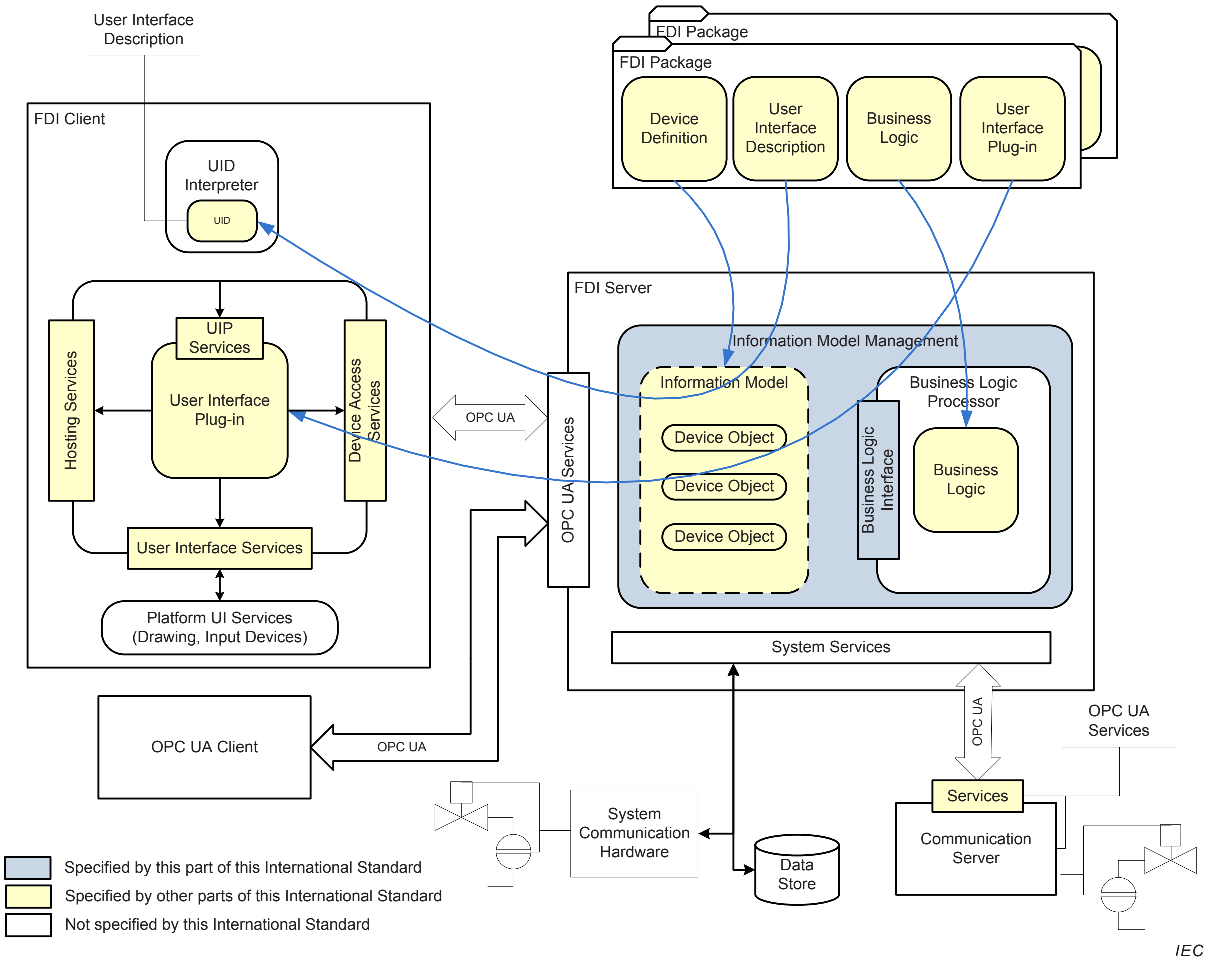

Figure 1 – FDI architecture diagram

## 2 Normative references

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

IEC 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 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 – Part 1: Overview*

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

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

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

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

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

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

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

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

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

## 3 Terms, definitions, abbreviated terms and acronyms

### 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.

**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 and acronyms

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

## 4 Overview

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

[1] To be published.

[2] To be published.

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.

## 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 built-in functions that shall be implemented by the FDI Server. The built-in functions that shall be implemented by the FDI Server are specified in IEC 61804.

## 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.

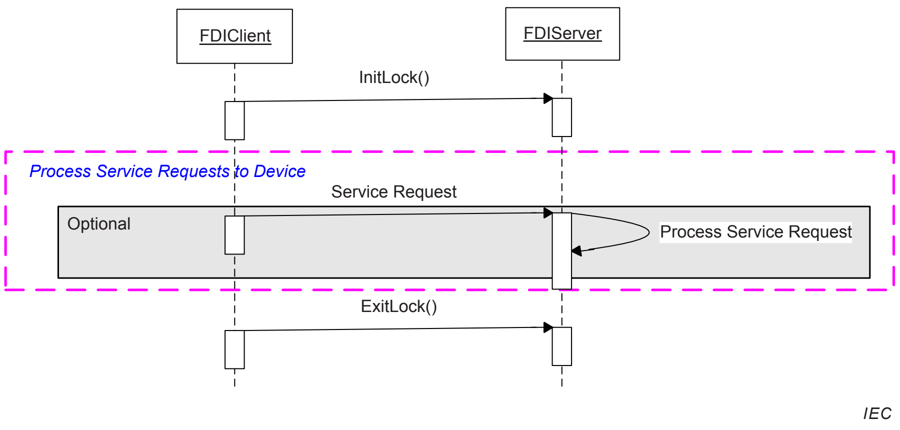

**Figure 2 – Locking services**

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 like 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.

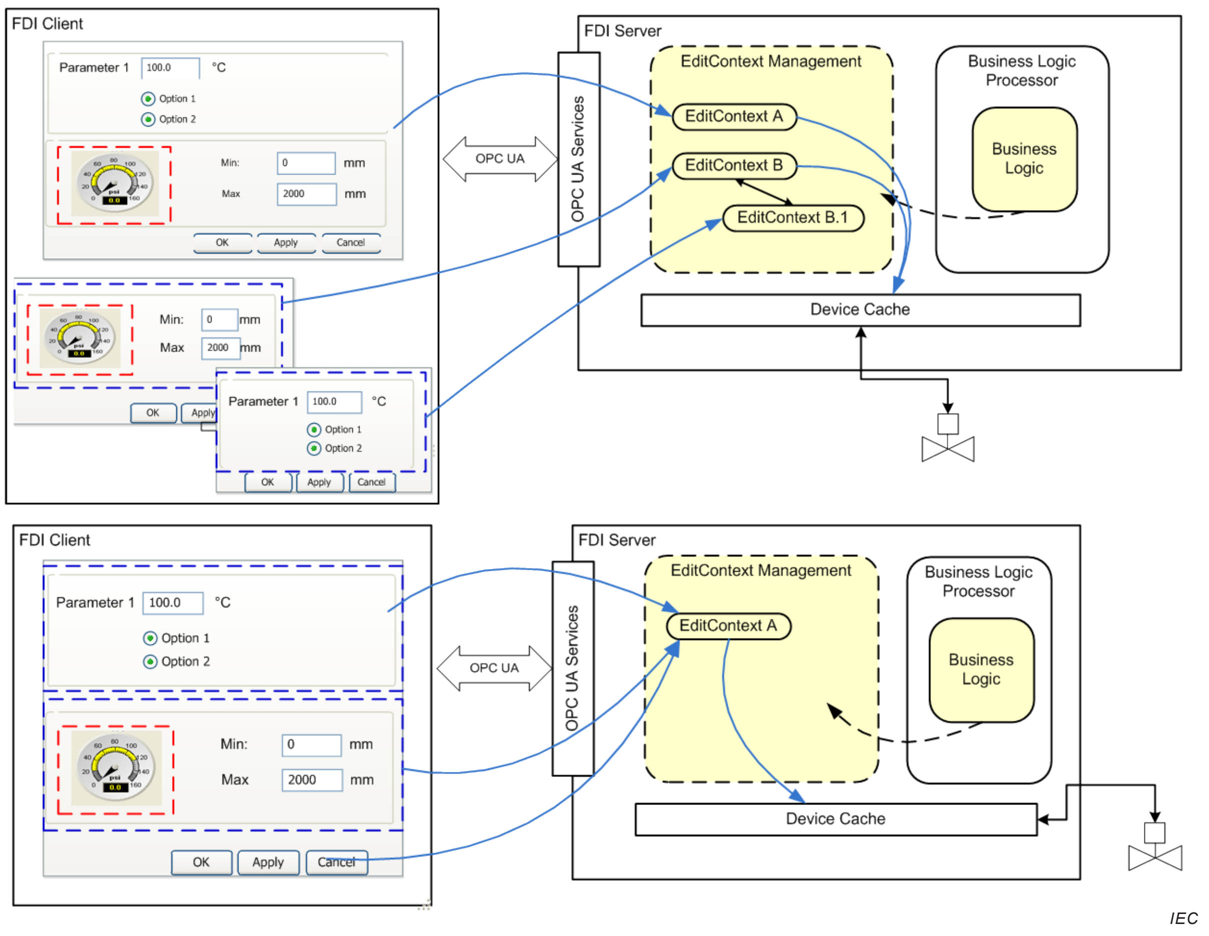

**Figure 3 – EditContext models**

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.

### 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.

### 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

### 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 4.

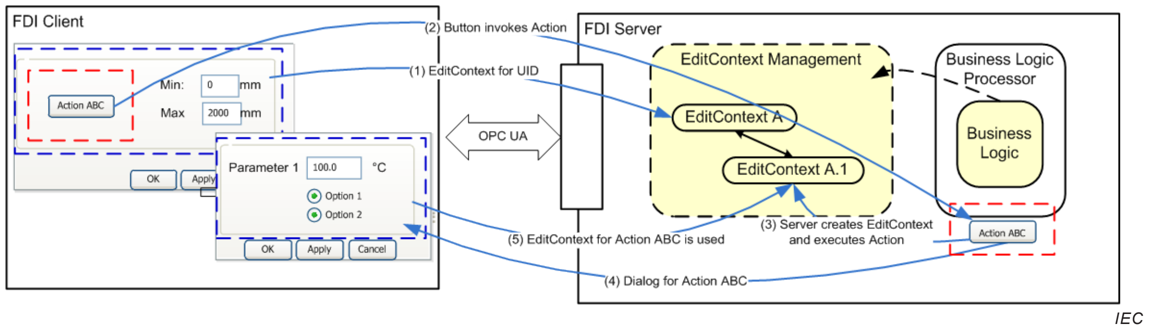

**Figure 4 – EditContext for EDD Methods**

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 built-in 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 built-in 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 5 shows the behavior of the FDI Server when an offline value is read.

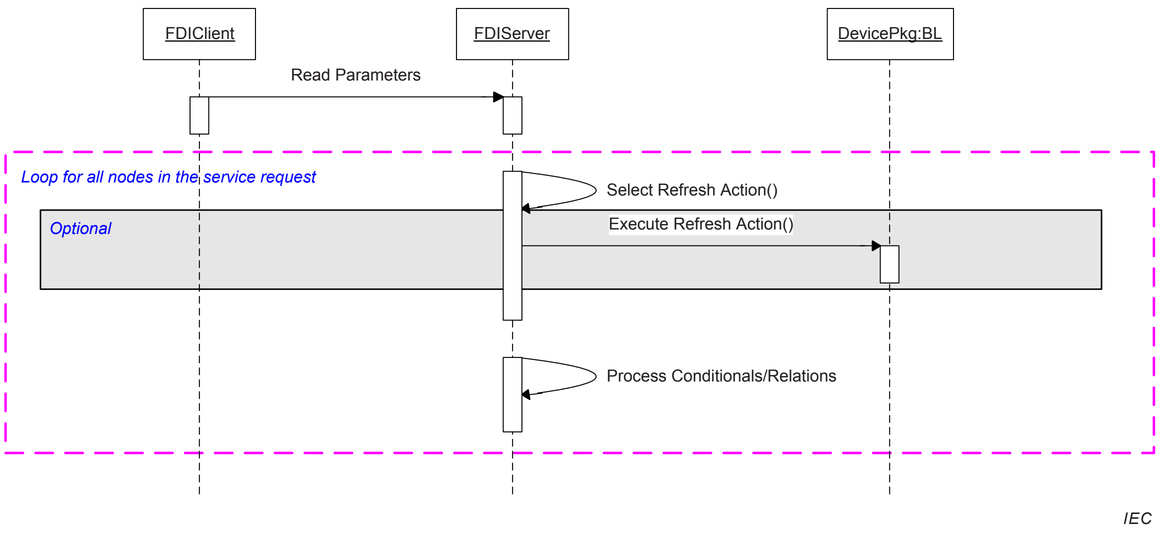

**Figure 5 – Offline variable read**

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 6 shows the behavior of the FDI Server when an online value is read.

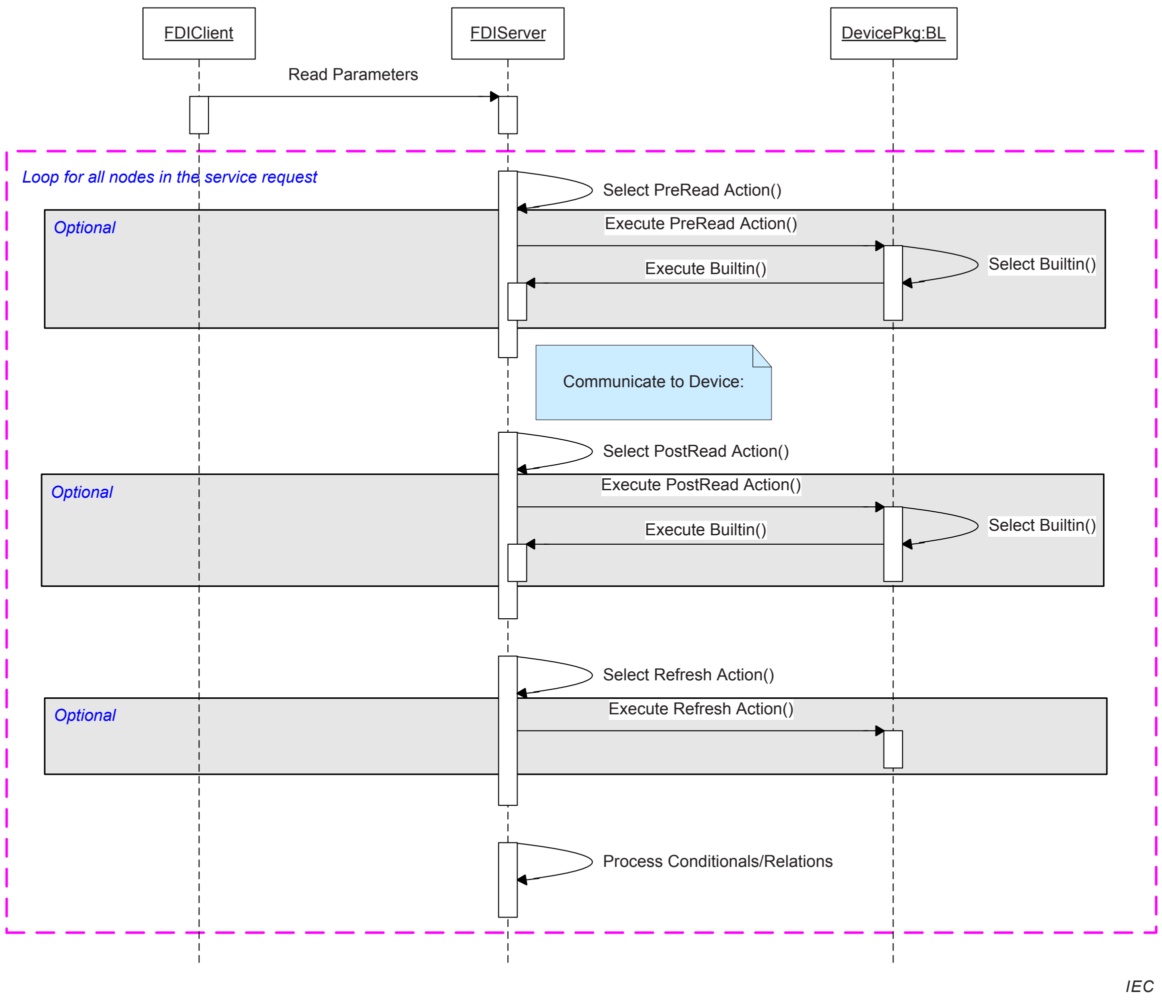

**Figure 6 – Online variable read**

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 built-in 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 7 shows the behavior of the FDI Server when an offline value is written.

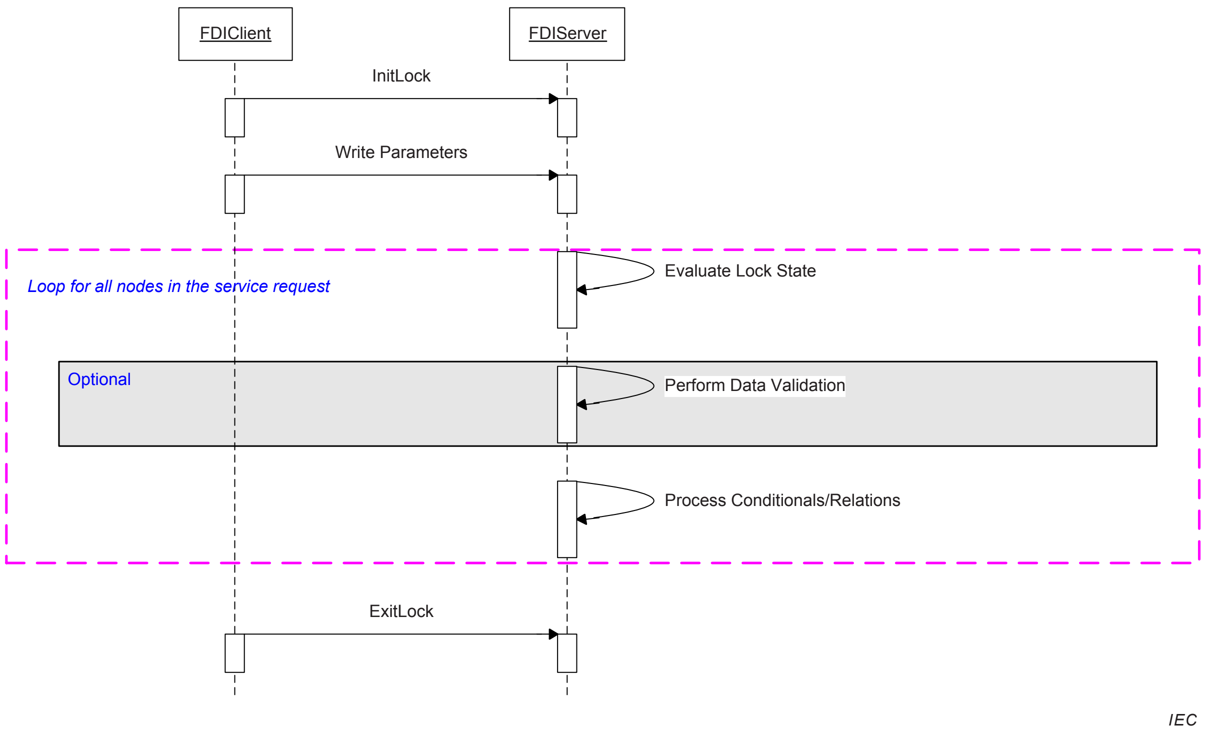

**Figure 7 – Offline variable write immediate**

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 8 shows the behavior of the FDI Server when an online value is written.

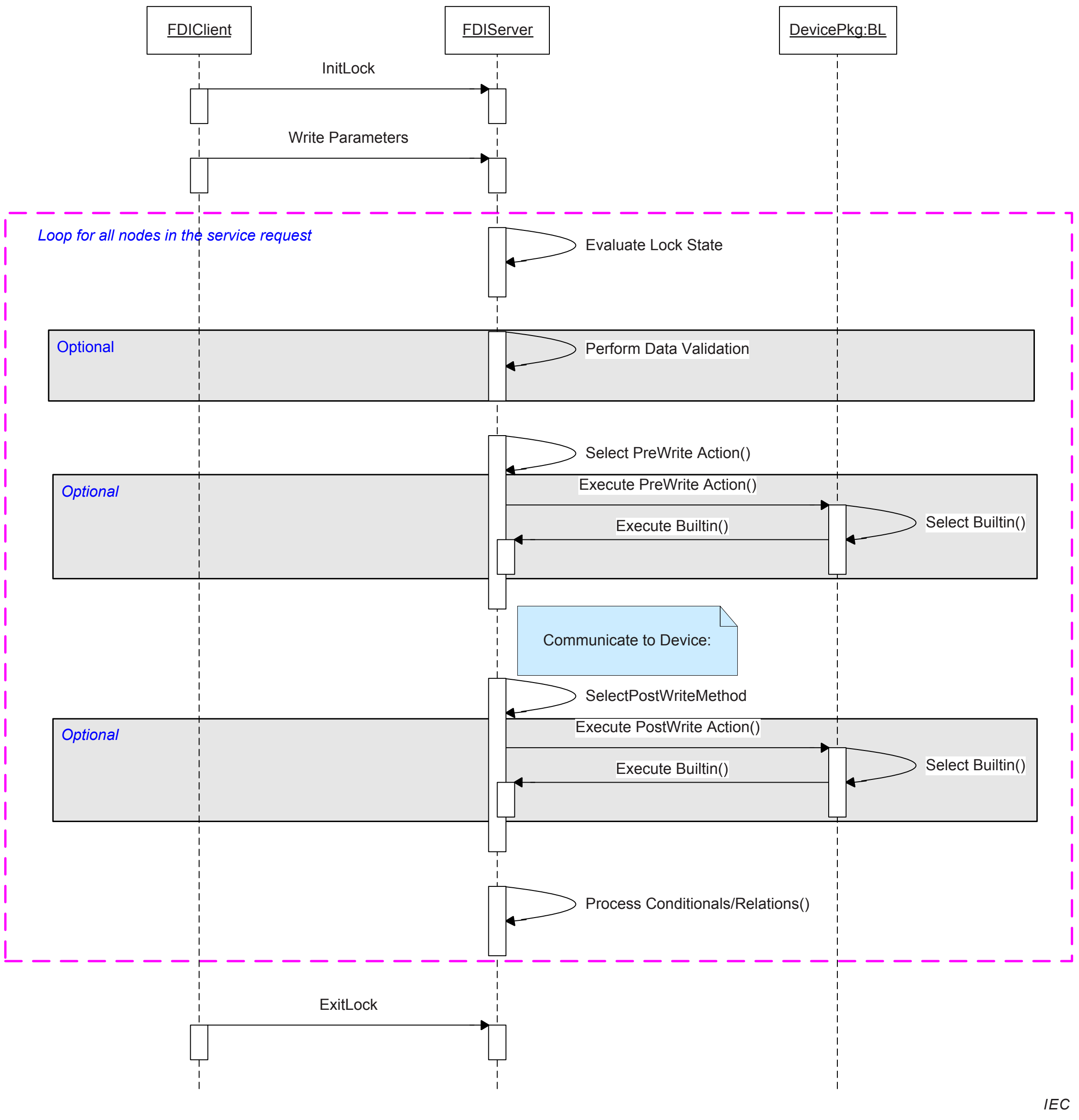

**Figure 8 – Online variable write immediate**

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.

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 9 shows the general behavior of an EditContext when Values are edited and applied.

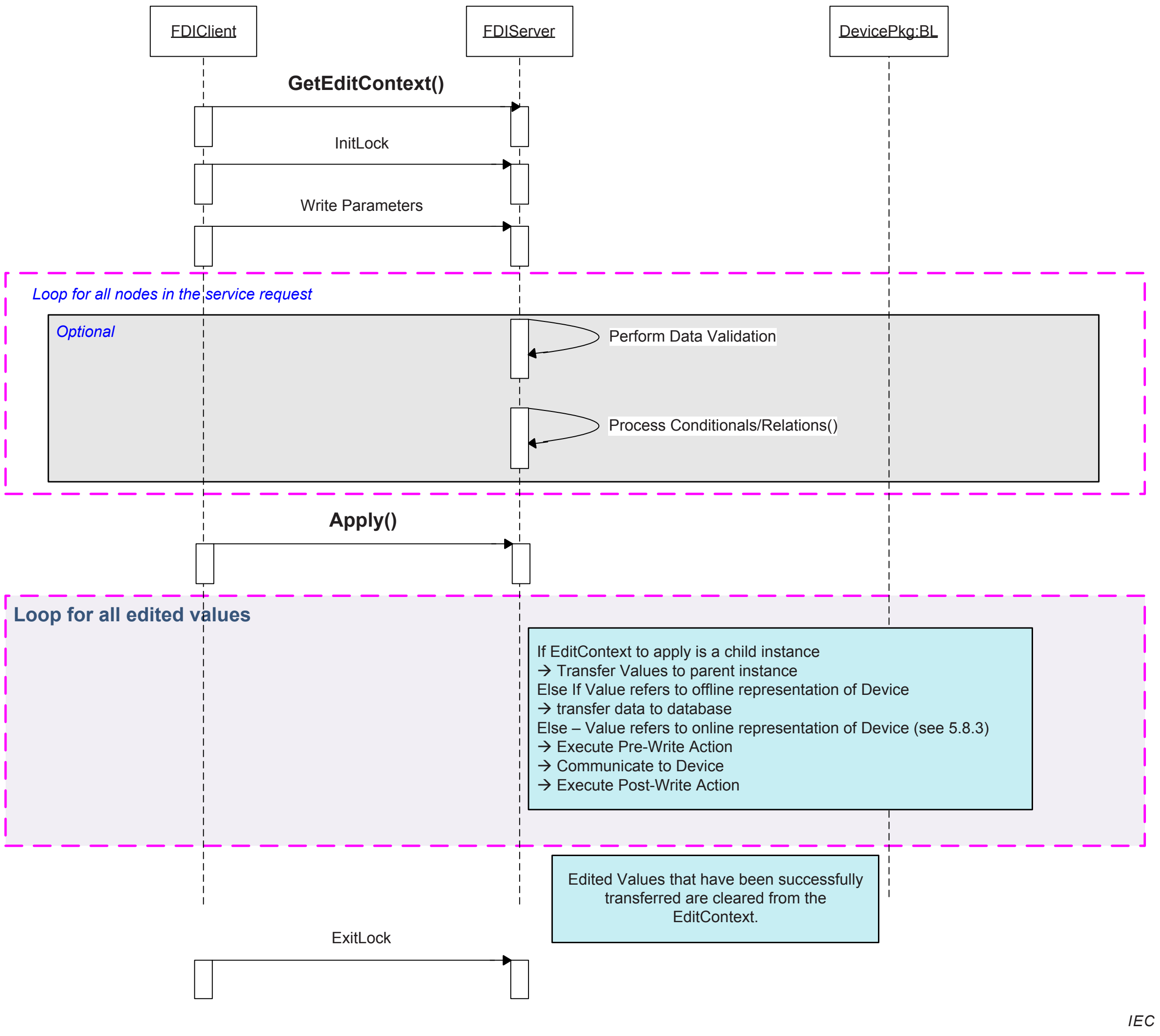
FDIClient
FDIServer
DevicePkg:BL
GetEditContext()
InitLock
Write Parameters
Loop for all nodes in the service request
Optional
Perform Data Validation
Process Conditionals/Relations()
Apply()
Loop for all edited values
If EditContext to apply is a child instance
→ Transfer Values to parent instance
Else If Value refers to offline representation of Device
→ transfer data to database
Else – Value refers to online representation of Device (see 5.8.3)
→ Execute Pre-Write Action
→ Communicate to Device
→ Execute Post-Write Action
Edited Values that have been successfully transferred are cleared from the EditContext.
ExitLock
IEC

**Figure 9 – 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 built-in 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 built-in 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. 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 10 shows the behavior of the FDI Server when an offline value is being monitored.

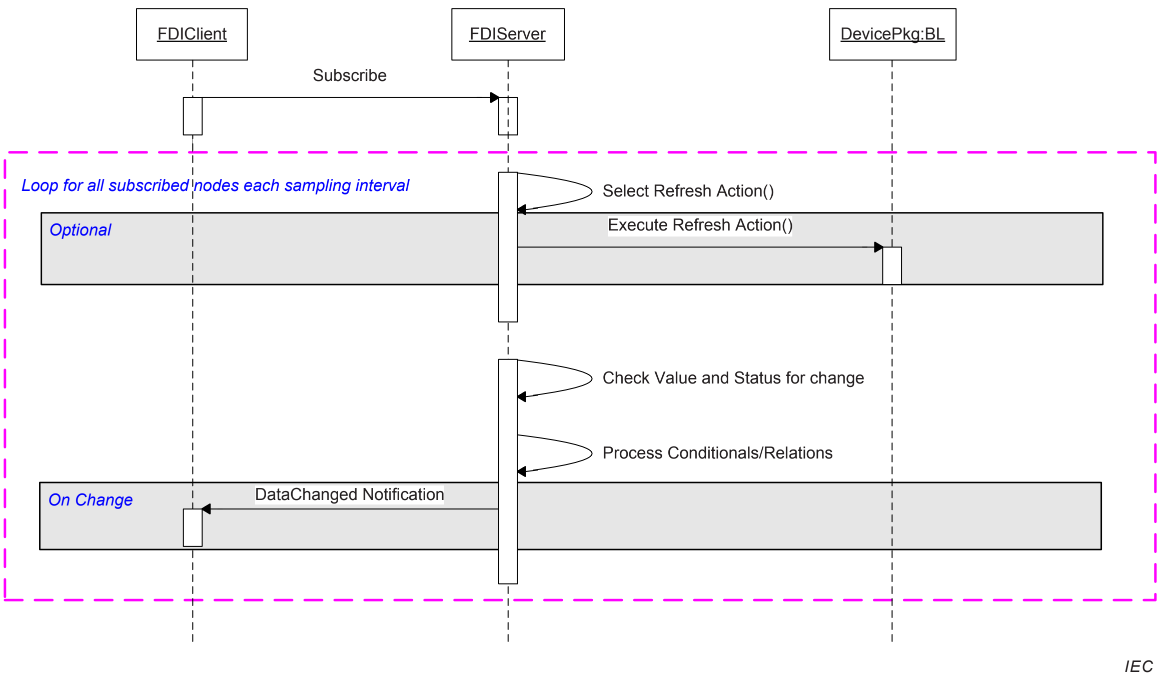

**Figure 10 – Offline variable subscription**

If a variable has refresh actions associated with it the FDI Server executes those actions 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 11 shows the behavior of the FDI Server when an online variable is being monitored.

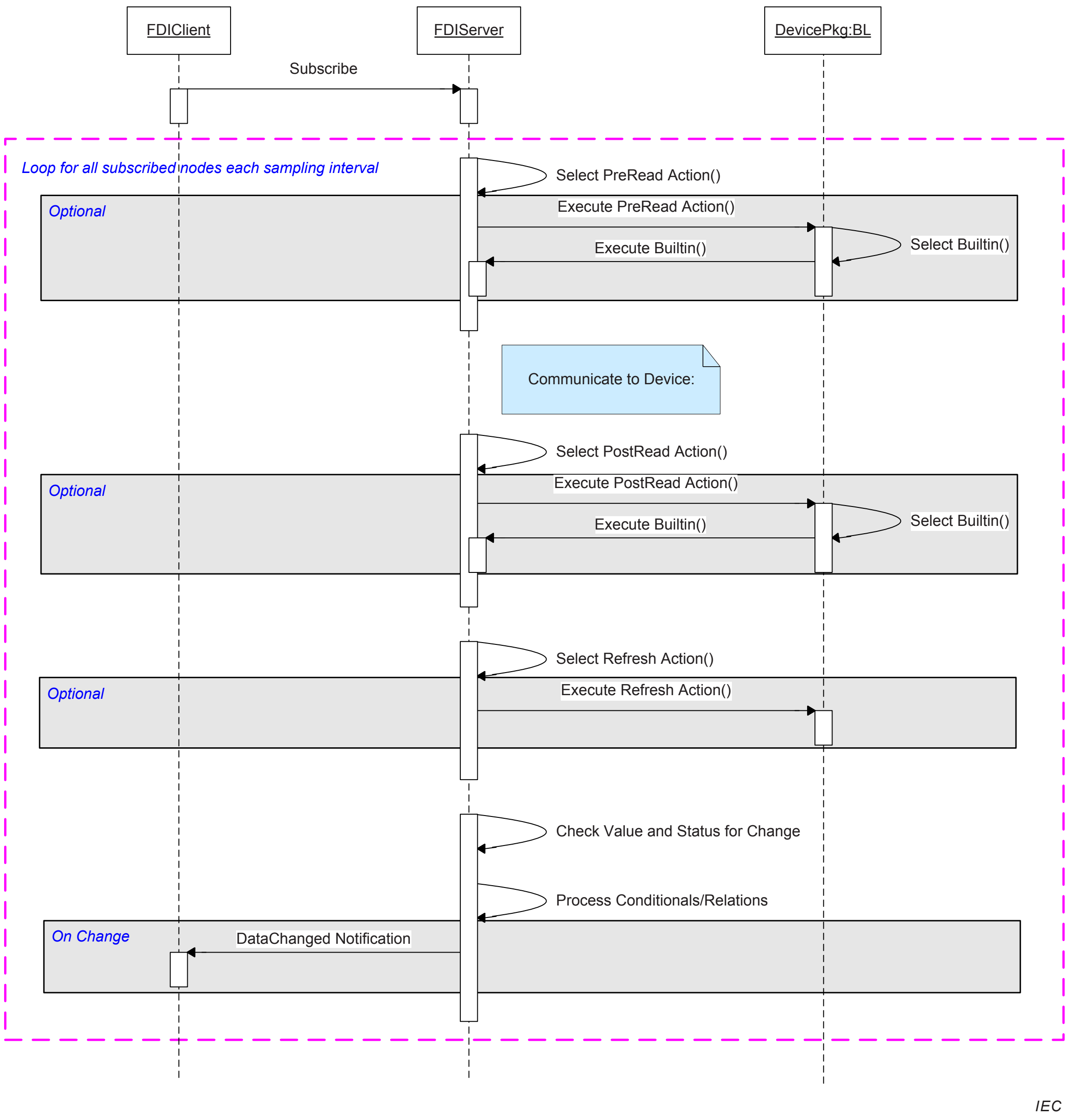

**Figure 11 – Online variable subscription**

The variable is read from the device 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 12 illustrates the topology within the Information Model.

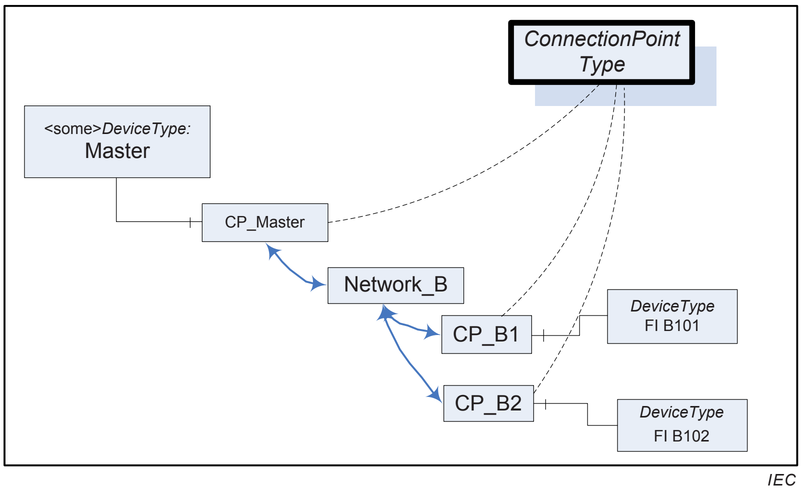

**Figure 12 – Topology with Network objects (non-normative)**

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 12). 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 13 shows the behavior of the FDI Server when a device is added to a network.

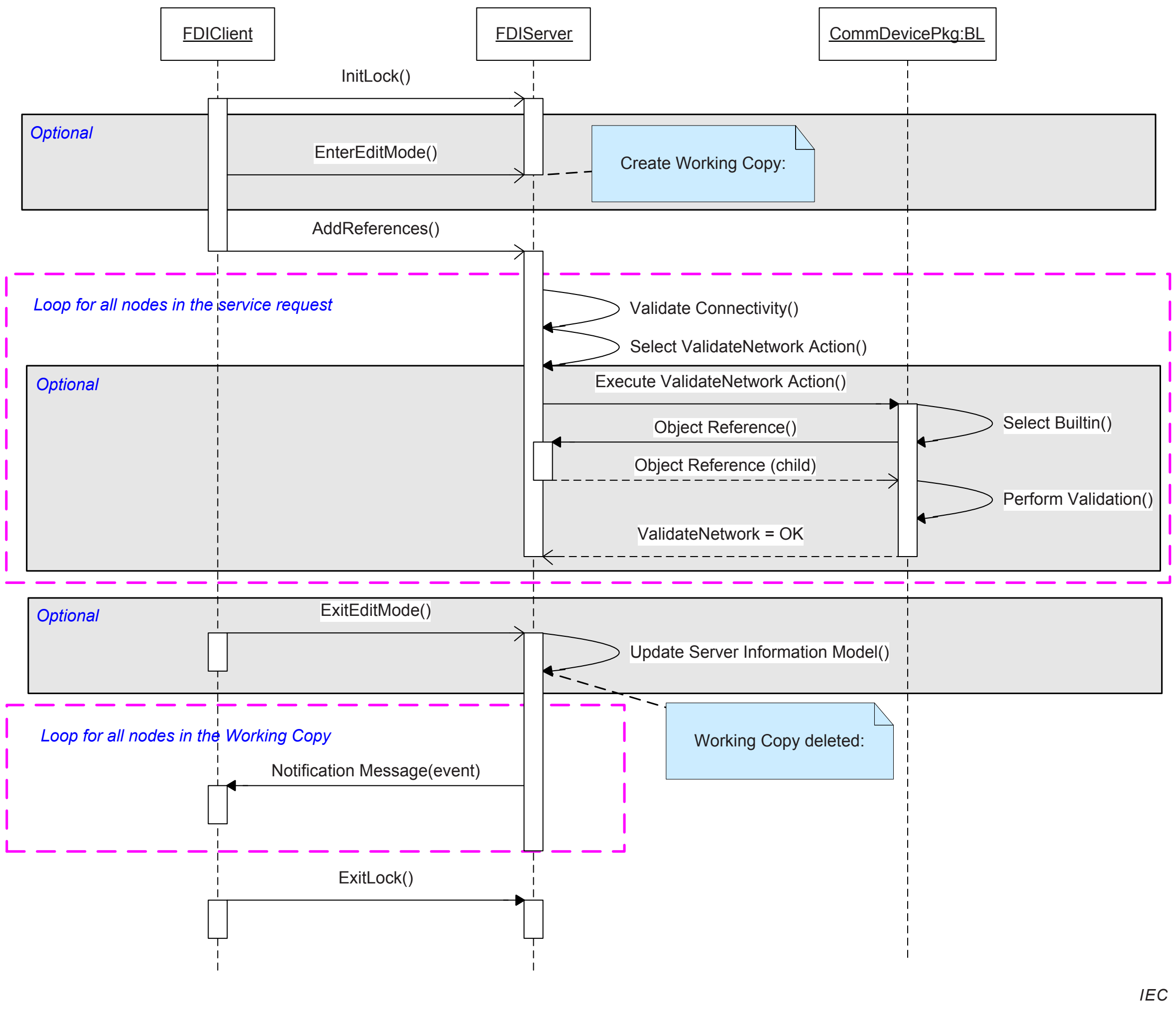

**Figure 13 – Add Device to topology**

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 14 shows the behavior of the FDI Server when a device is removed from a network.

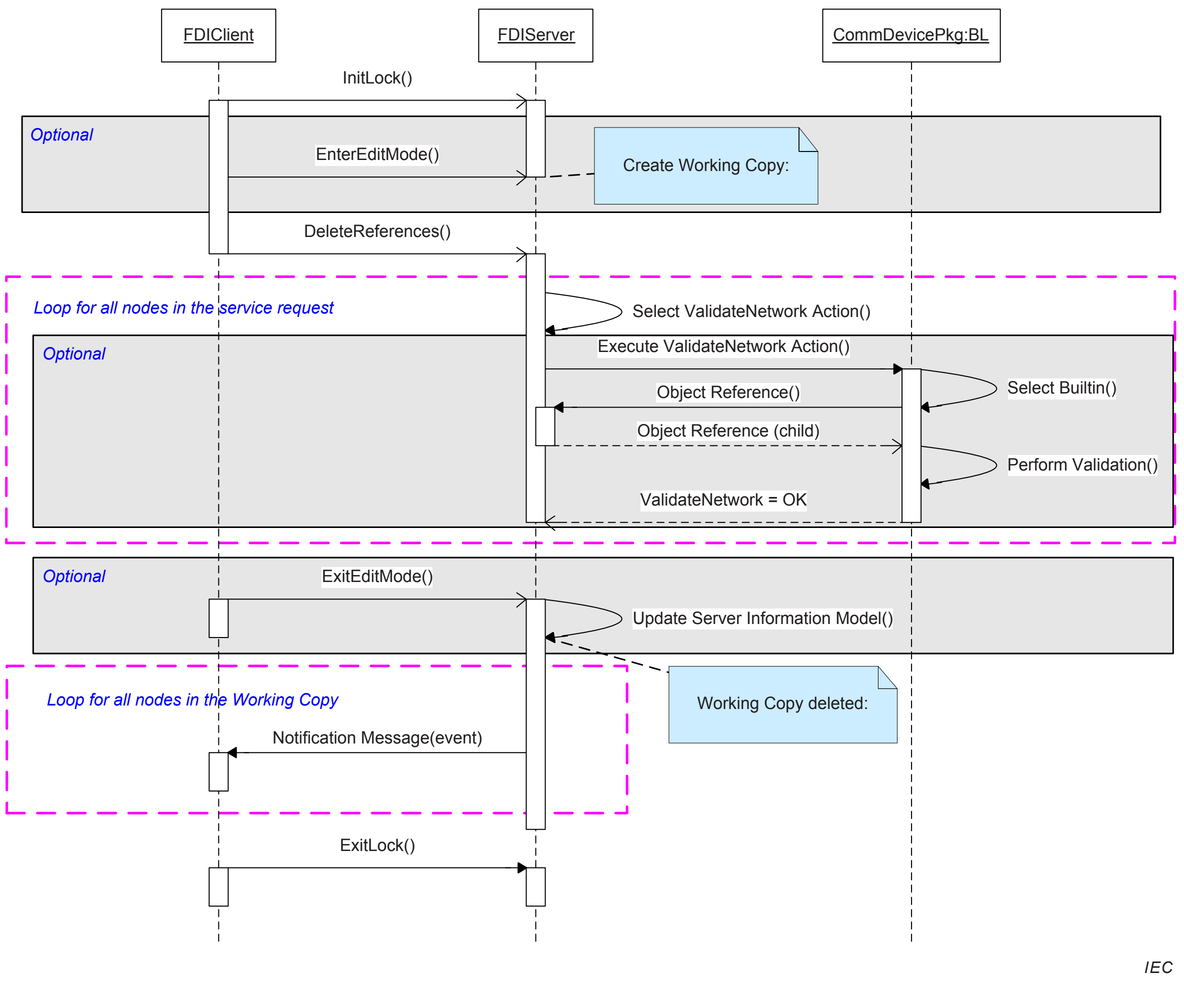

**Figure 14 – Remove Device from topology**

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 15 shows the behavior of the FDI Server when the topology is scanned.

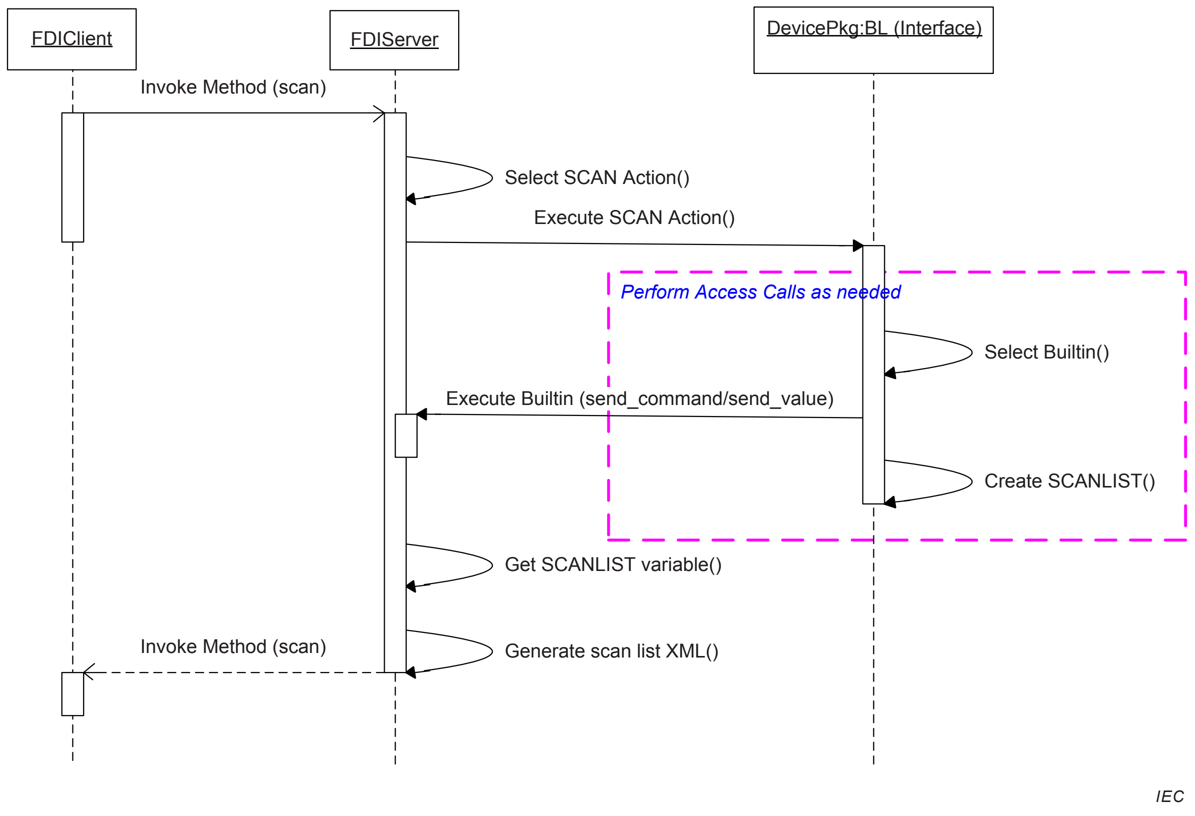

**Figure 15 – Scan topology**

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 built-ins 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. 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. 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 (seeIEC 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 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 built-in 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 16 shows the client/server interaction of an Action.

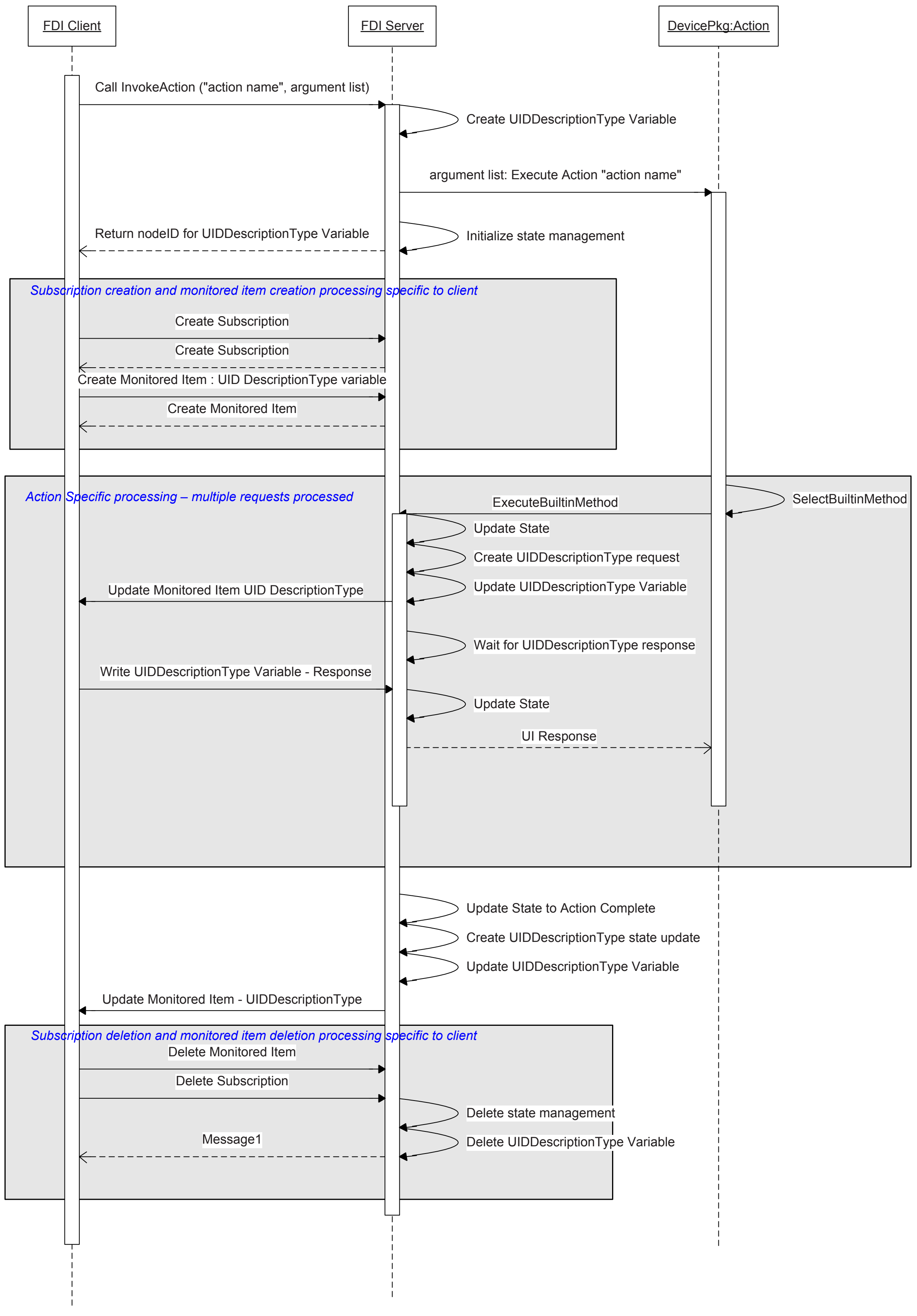
FDI Client
FDI Server
DevicePkg:Action
Call InvokeAction ("action name", argument list)
Create UIDDescriptionType Variable
argument list: Execute Action "action name"
Return nodeID for UIDDescriptionType Variable
Initialize state management
Subscription creation and monitored item creation processing specific to client
Create Subscription
Create Subscription
Create Monitored Item : UID DescriptionType variable
Create Monitored Item
Action Specific processing – multiple requests processed
ExecuteBuiltinMethod
SelectBuiltinMethod
Update State
Create UIDDescriptionType request
Update Monitored Item UID DescriptionType
Update UIDDescriptionType Variable
Wait for UIDDescriptionType response
Write UIDDescriptionType Variable - Response
Update State
UI Response
Update State to Action Complete
Create UIDDescriptionType state update
Update UIDDescriptionType Variable
Update Monitored Item - UIDDescriptionType
Subscription deletion and monitored item deletion processing specific to client
Delete Monitored Item
Delete Subscription
Delete state management
Message1
Delete UIDDescriptionType Variable
IEC

**Figure 16 – Action execution**

### 5.12.2 Action state machine

#### 5.12.2.1 States

The Action state machine is shown in Figure 17.

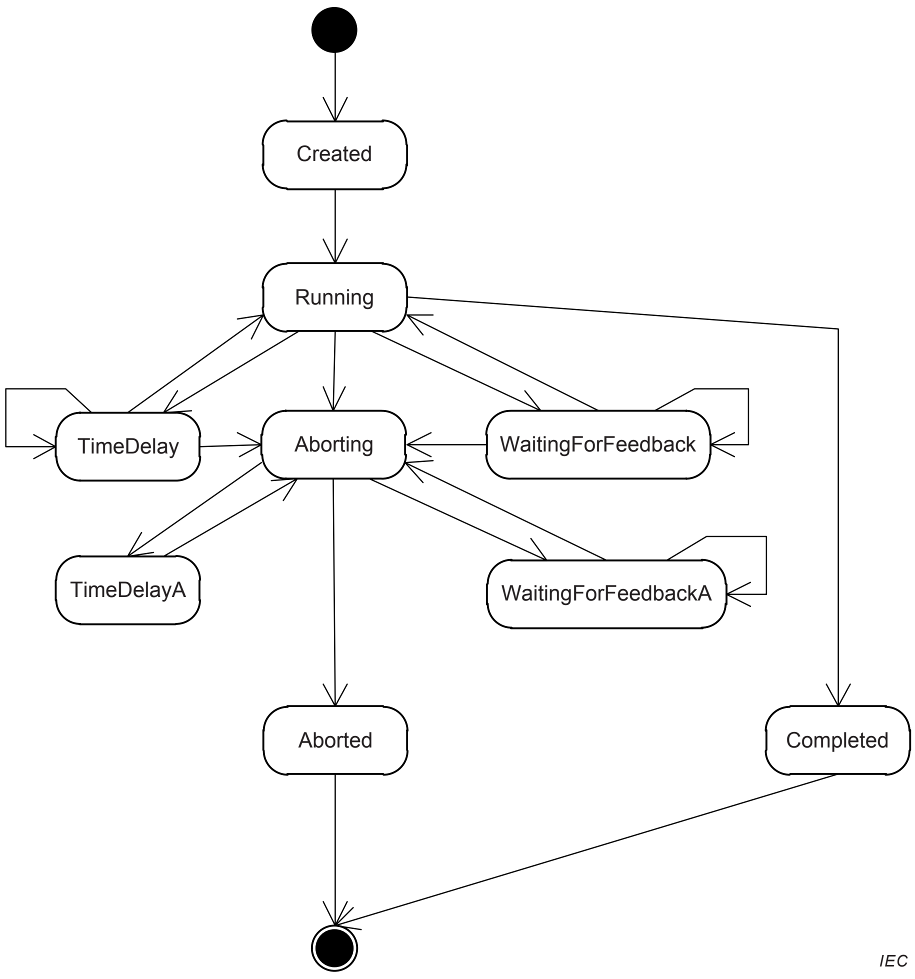

**Figure 17 – Action state machine**

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 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 a certain amount of time. |
| WaitingForFeedbackA | The state where the abort processing is suspended because a user interaction is needed. |
| 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 | Built-in function has been encountered that requires a time delay. | TimeDelay |
| Running | Built-in 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 | Built-in function has been encountered that requires a time delay. | TimeDelayA |
| Aborting | Built-in 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, 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 mean time 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 built-in 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 according to 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 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 scope.

Figure 18 shows a possible architecture example of system communication integration.

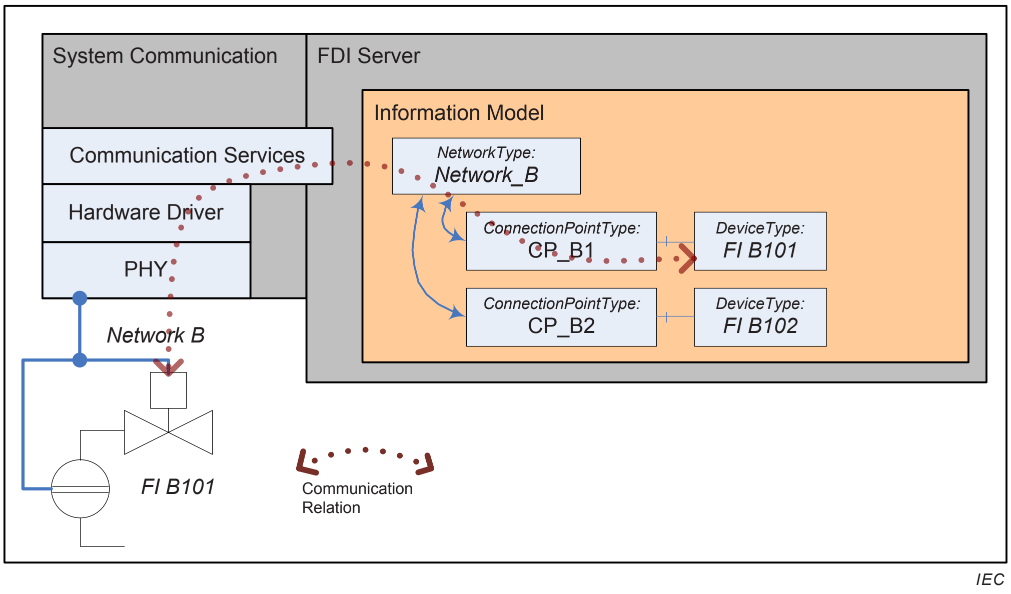

**Figure 18 – System communication integration example**

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 19.

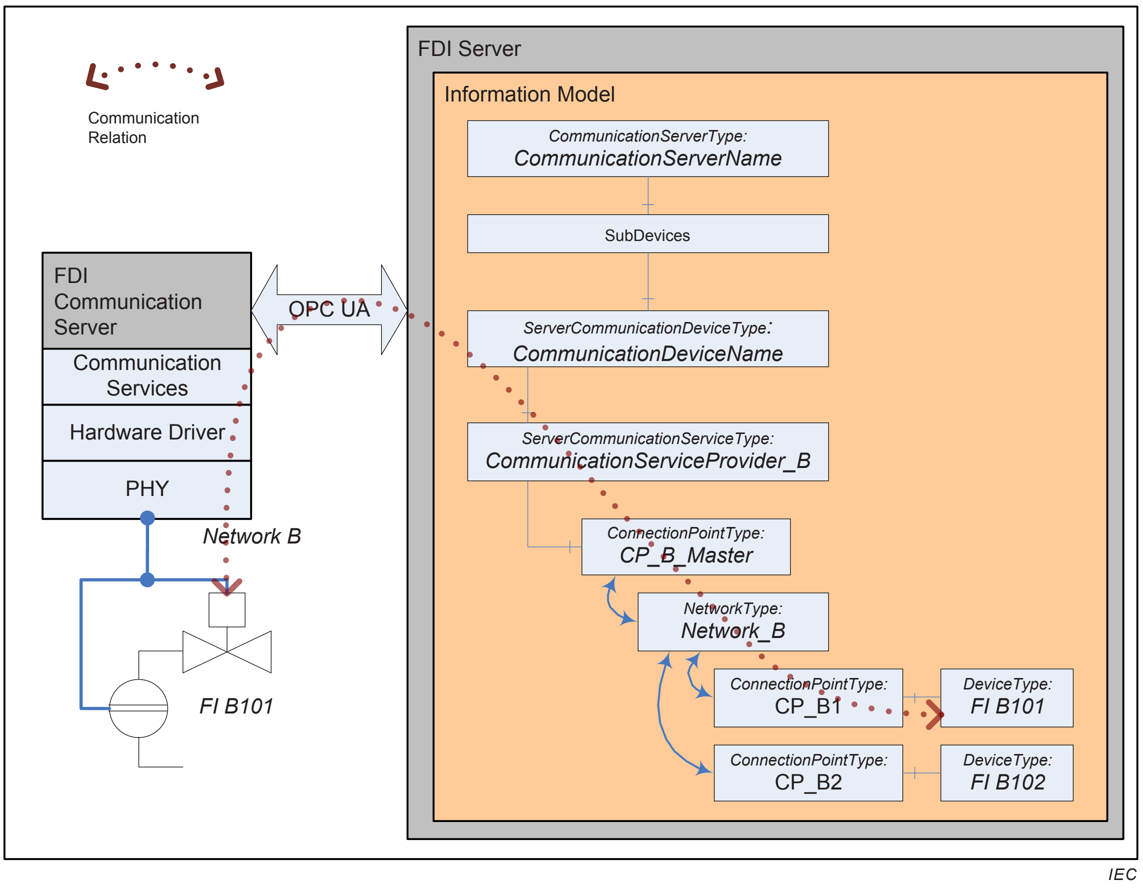

**Figure 19 – FDI Communication Server integration example**

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 20 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 20.

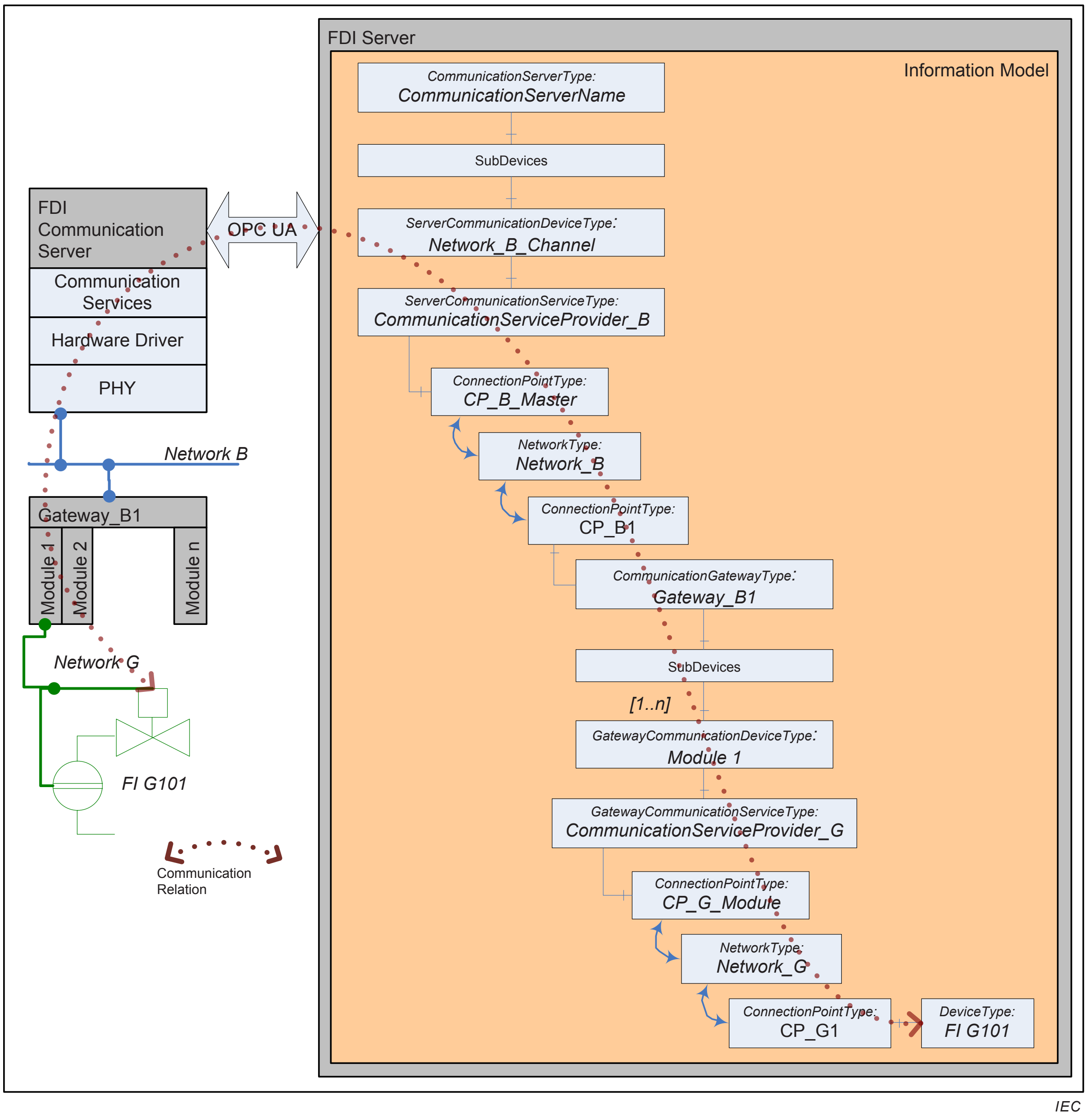

**Figure 20 – Gateway integration example**

### 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 (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 (IEC 62769-5).
6) FDI Communication Server (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 20 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 20. 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 standard.

### 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 according to 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 Built-In 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 according to 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 21 represents an example scenario derived from Figure 20.

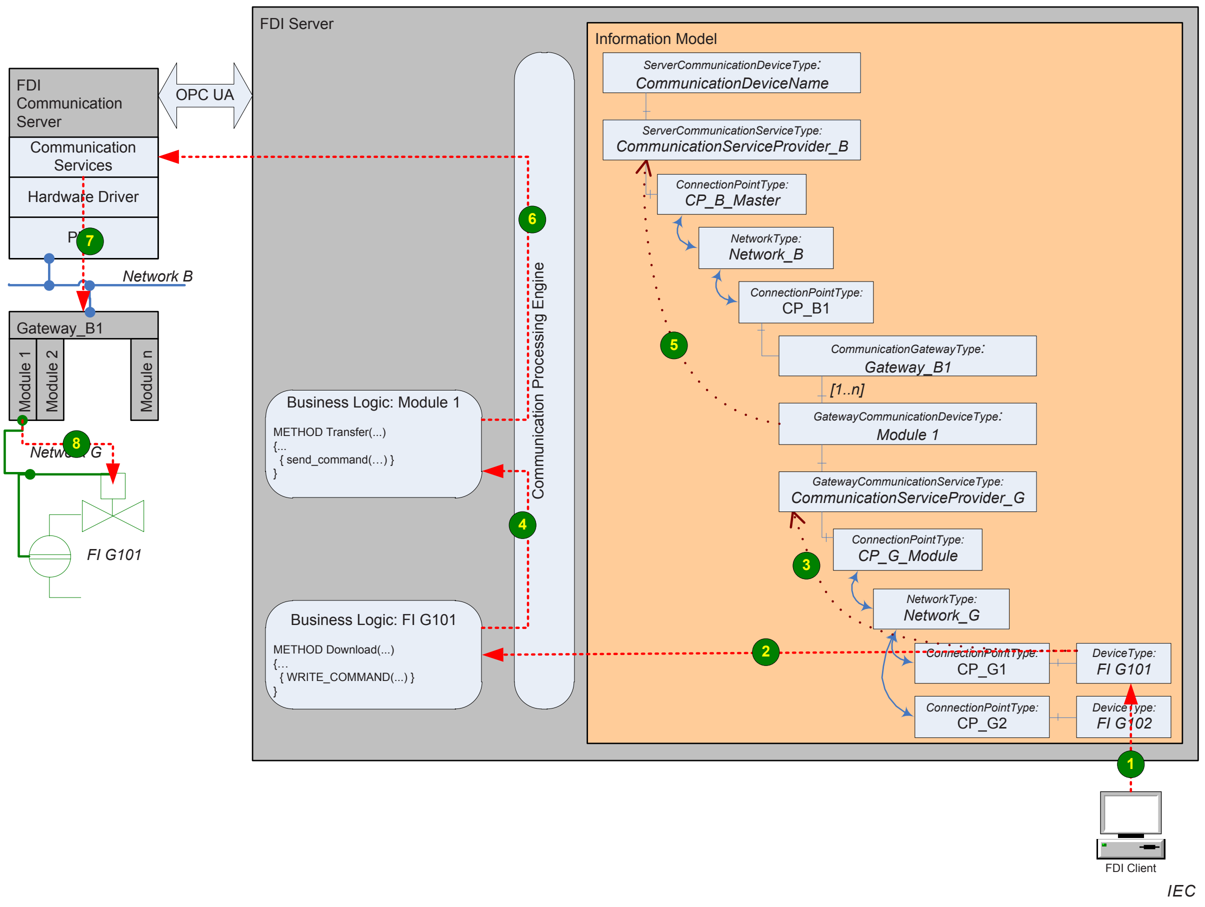
FDI Server
Information Model
FDI Communication Server
Communication Services
Hardware Driver
OPC UA
Network B
Gateway_B1
Module 1
Module 2
Module n
Network G
FI G101
Communication Processing Engine
Business Logic: Module 1
METHOD Transfer(...)
{...
{ send_command(...) }
}
Business Logic: FI G101
METHOD Download(...)
{...
{ WRITE_COMMAND(...) }
}
ServerCommunicationDeviceType: CommunicationDeviceName
ServerCommunicationServiceType: CommunicationServiceProvider_B
ConnectionPointType: CP_B_Master
NetworkType: Network_B
ConnectionPointType: CP_B1
CommunicationGatewayType: Gateway_B1
[1..n]
GatewayCommunicationDeviceType: Module 1
GatewayCommunicationServiceType: CommunicationServiceProvider_G
ConnectionPointType: CP_G_Module
NetworkType: Network_G
ConnectionPointType: CP_G1
ConnectionPointType: CP_G2
DeviceType: FI G101
DeviceType: FI G102
FDI Client
1
2
3
4
5
6
7
8
IEC

**Figure 21 – Message propagation example scenario**

NOTE 1 The numbers in brackets (1) to (8) used in the the following description refer to elements in Figure 21.

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 Built-In 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 Built-In 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 Built-In, 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 built-ins 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

According to IEC 62769-7 the FDI Communication Package contains an EDD element describing the VARIABLES and Business Logic mapped in to 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 follow the rules given below:

a) An activity at an EDD entity is always a consequence of a trigger.
b) An activity at an EDD entity cannot be interrupted.
c) An activity runs until the activity is finished or aborted.
d) An EDD entity can only execute one activity at a time.
e) 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.
f) 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).
g) 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.
h) 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.
i) The server can resolve deadlock scenarios by aborting one of the involved activity chains.

# 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.

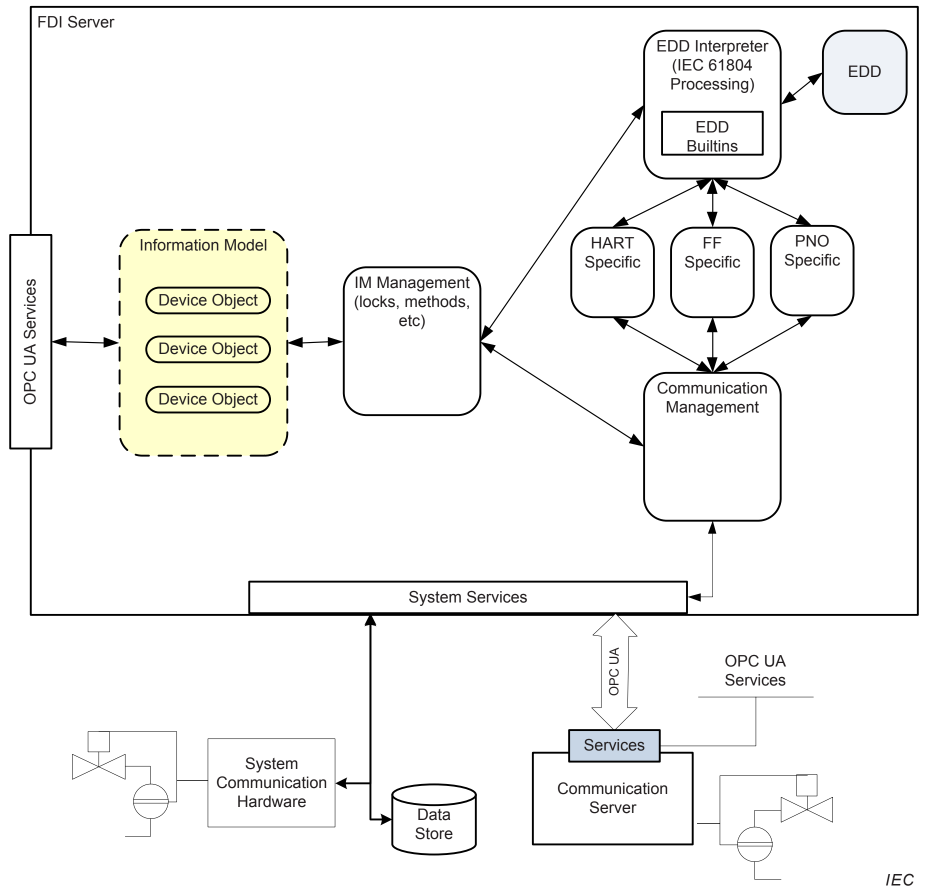

**Figure A.1 – Functional components of an FDI Server**

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.

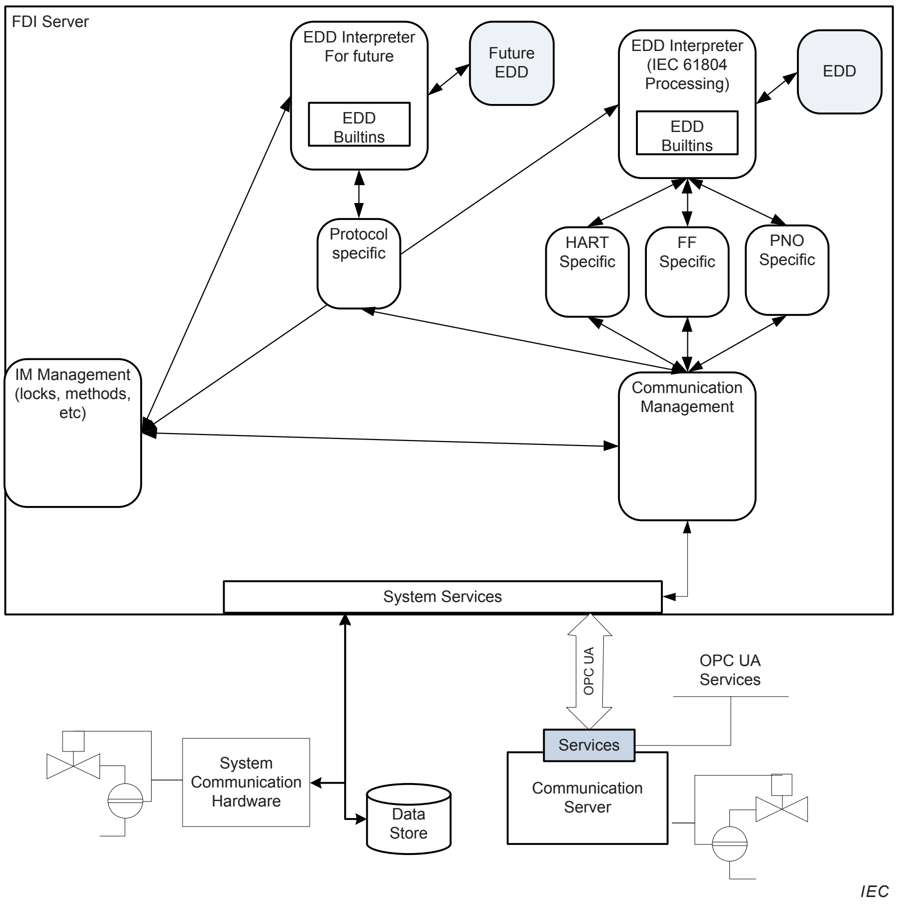

Figure A.2 – FDI Server extensions

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.

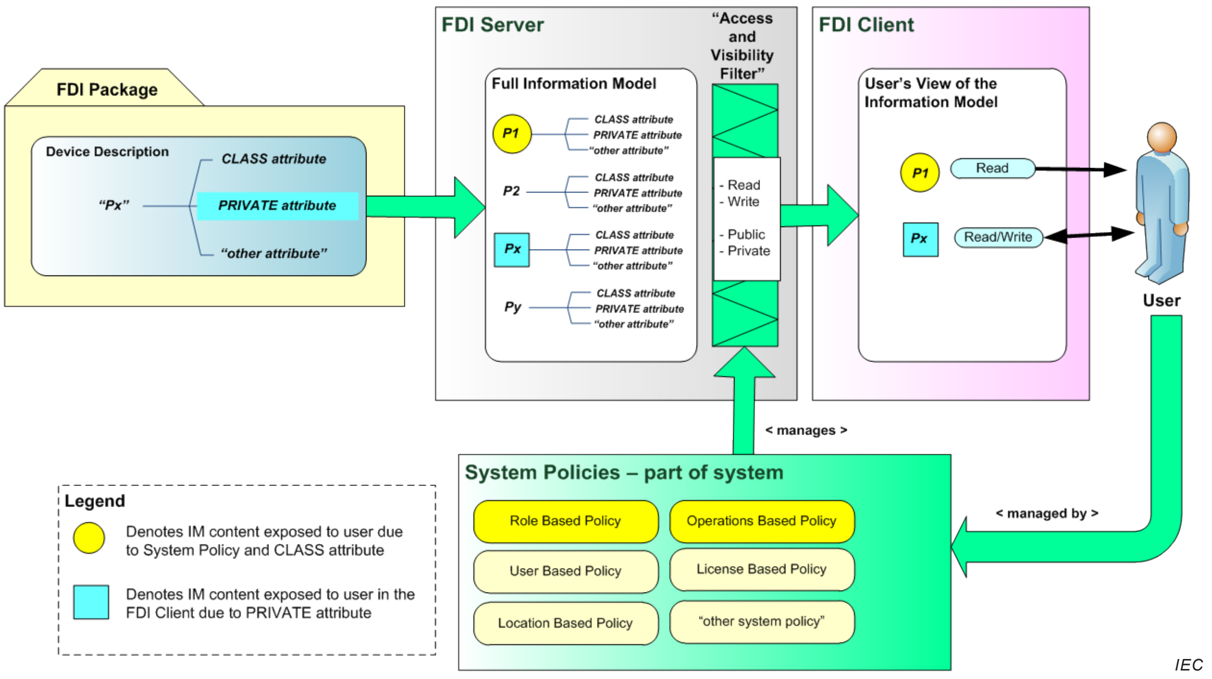

**Figure B.1 – User roles and access privileges**

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.

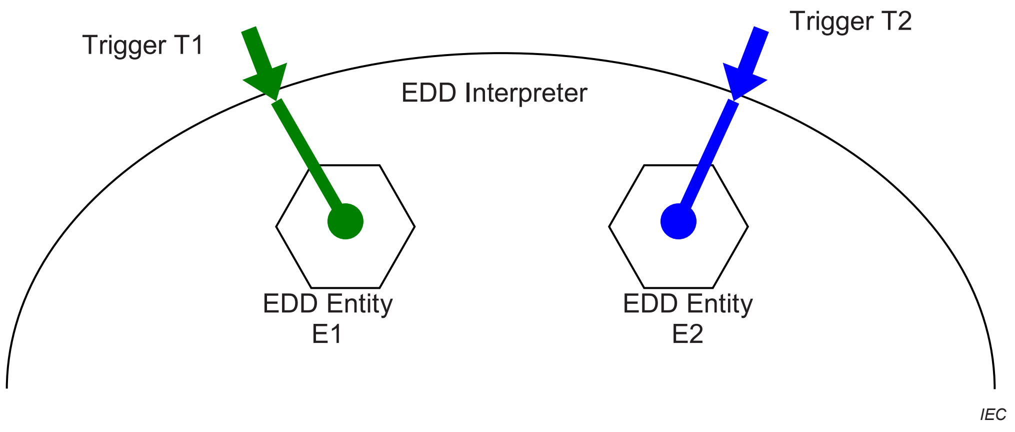

**Figure C.1 – Synchronous execution of two triggers**

## 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.

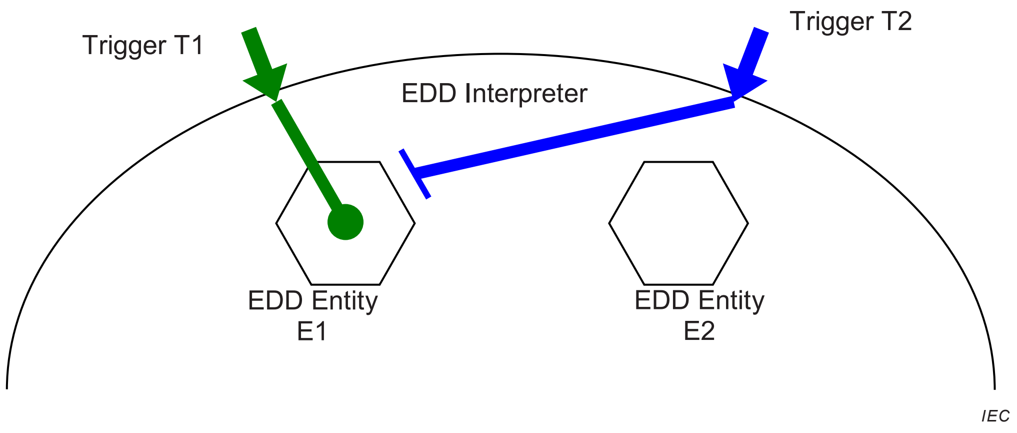

**Figure C.2 – Concurrent execution of two triggers (step1)**

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.

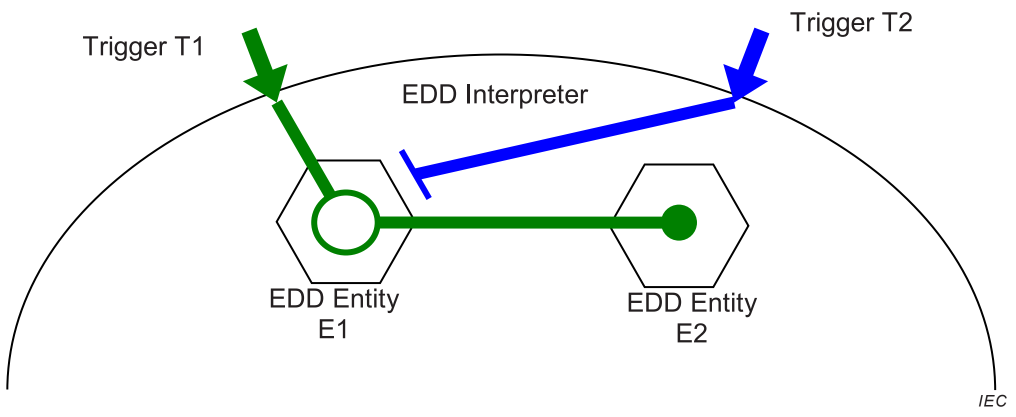

**Figure C.3 – Concurrent execution of two triggers (step 2)**

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.

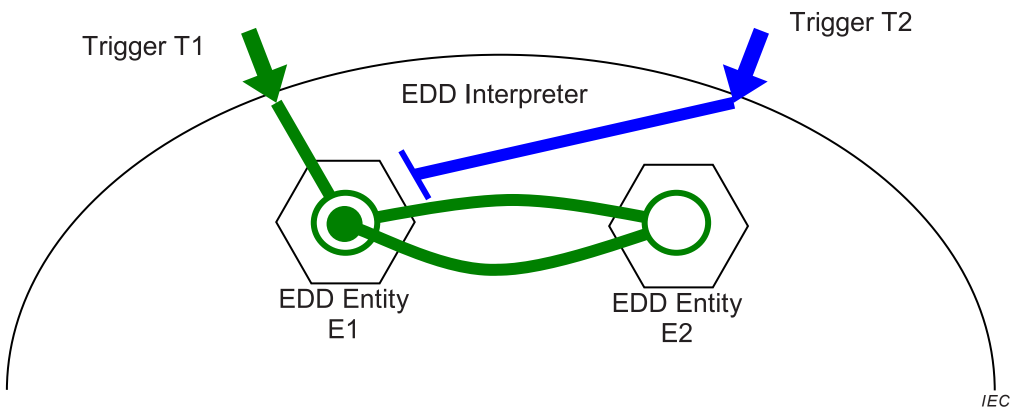

**Figure C.4 – Concurrent execution of two triggers (step 3)**

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.

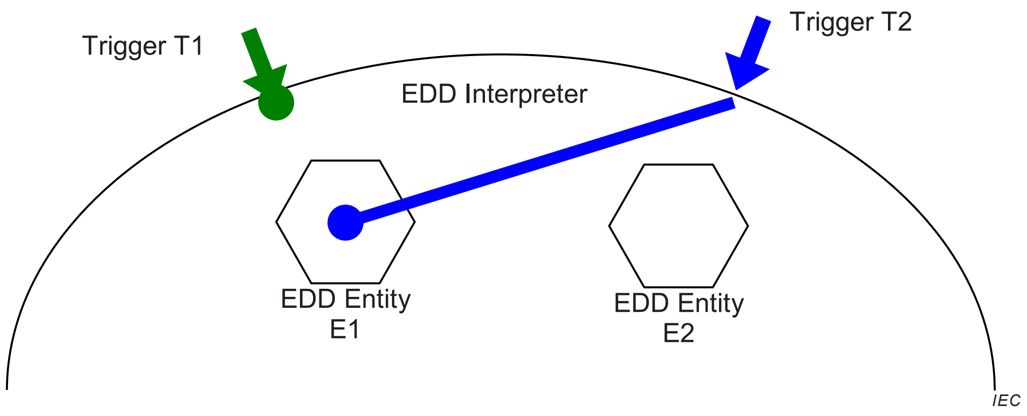

**Figure C.5 – Concurrent execution of two triggers (step 4)**

## 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.

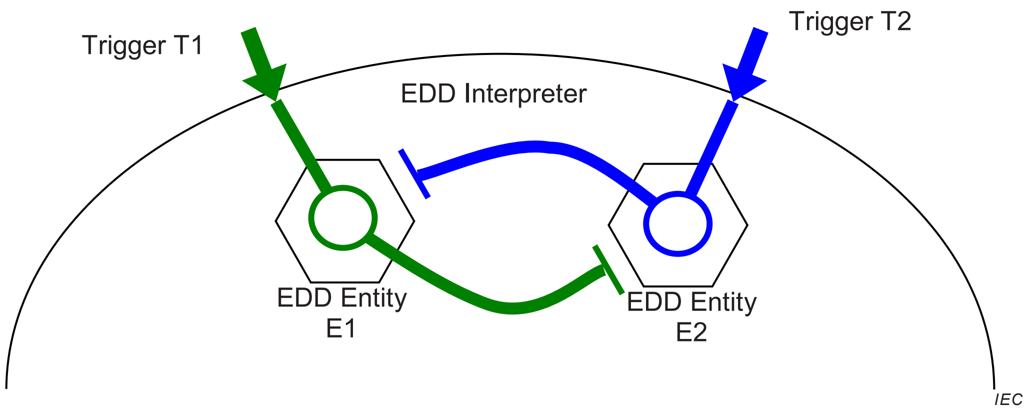

**Figure C.6 – Concurrent execution of two triggers**

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.

## Bibliography

IEC 61804-5, *Function blocks (FB) for process control and Electronic Device Description Language (EDDL) – Part 5: EDDL Builtin library*

IEC 62769-6, *Field Device Integration (FDI) – Part 6: FDI Technology Mapping*

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

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

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

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

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

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

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

___________

*This page deliberately left blank*

# British Standards Institution (BSI)

BSI is the national body responsible for preparing British Standards and other standards-related publications, information and services.

BSI is incorporated by Royal Charter. British Standards and other standardization products are published by BSI Standards Limited.

## About us

We bring together business, industry, government, consumers, innovators and others to shape their combined experience and expertise into standards -based solutions.

The knowledge embodied in our standards has been carefully assembled in a dependable format and refined through our open consultation process. Organizations of all sizes and across all sectors choose standards to help them achieve their goals.

## Information on standards

We can provide you with the knowledge that your organization needs to succeed. Find out more about British Standards by visiting our website at bsigroup.com/standards or contacting our Customer Services team or Knowledge Centre.

## Buying standards

You can buy and download PDF versions of BSI publications, including British and adopted European and international standards, through our website at bsigroup.com/shop, where hard copies can also be purchased.

If you need international and foreign standards from other Standards Development Organizations, hard copies can be ordered from our Customer Services team.

## Subscriptions

Our range of subscription services are designed to make using standards easier for you. For further information on our subscription products go to bsigroup.com/subscriptions.

With **British Standards Online (BSOL)** you'll have instant access to over 55,000 British and adopted European and international standards from your desktop. It's available 24/7 and is refreshed daily so you'll always be up to date.

You can keep in touch with standards developments and receive substantial discounts on the purchase price of standards, both in single copy and subscription format, by becoming a **BSI Subscribing Member**.

**PLUS** is an updating service exclusive to BSI Subscribing Members. You will automatically receive the latest hard copy of your standards when they're revised or replaced.

To find out more about becoming a BSI Subscribing Member and the benefits of membership, please visit bsigroup.com/shop.

With a **Multi-User Network Licence (MUNL)** you are able to host standards publications on your intranet. Licences can cover as few or as many users as you wish. With updates supplied as soon as they're available, you can be sure your documentation is current. For further information, email bsmusales@bsigroup.com.

## Revisions

Our British Standards and other publications are updated by amendment or revision.

We continually improve the quality of our products and services to benefit your business. If you find an inaccuracy or ambiguity within a British Standard or other BSI publication please inform the Knowledge Centre.

## Copyright

All the data, software and documentation set out in all British Standards and other BSI publications are the property of and copyrighted by BSI, or some person or entity that owns copyright in the information used (such as the international standardization bodies) and has formally licensed such information to BSI for commercial publication and use. Except as permitted under the Copyright, Designs and Patents Act 1988 no extract may be reproduced, stored in a retrieval system or transmitted in any form or by any means – electronic, photocopying, recording or otherwise – without prior written permission from BSI. Details and advice can be obtained from the Copyright & Licensing Department.

## Useful Contacts:

**Customer Services**
**Tel:** +44 845 086 9001
**Email (orders):** orders@bsigroup.com
**Email (enquiries):** cservices@bsigroup.com

**Subscriptions**
**Tel:** +44 845 086 9001
**Email:** subscriptions@bsigroup.com

**Knowledge Centre**
**Tel:** +44 20 8996 7004
**Email:** knowledgecentre@bsigroup.com

**Copyright & Licensing**
**Tel:** +44 20 8996 7070
**Email:** copyright@bsigroup.com

**BSI Group Headquarters**

389 Chiswick High Road London W4 4AL UK

bsi.

...making excellence a habit.™