IEC 62769-7

Edition 3.0 2023-04
REDLINE VERSION

INTERNATIONAL STANDARD

colour inside

Field Device Integration (FDI®) – Part 7: Communication Devices

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

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

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

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

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 It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished
Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and once a month by email.

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

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

Electropedia - www.electropedia.org
The world's leading online dictionary on electrotechnology, containing more than 22 300 terminological entries in English and French, with equivalent terms in 19 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC 62769-7

Edition 3.0 2023-04
REDLINE VERSION

INTERNATIONAL STANDARD

colour inside

Field Device Integration (FDI®) – Part 7: Communication Devices

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

ICS 25.040.40; 35.100.05

ISBN 978-2-8322-6842-1

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

FIELD DEVICE INTEGRATION (FDI®) –

Part 7: Communication Devices

FOREWORD

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

This redline version of the official IEC Standard allows the user to identify the changes made to the previous edition IEC 62769-7:2021. A vertical bar appears in the margin wherever a change has been made. Additions are in green text, deletions are in strikethrough red text.

IEC 62769-7 has been prepared by subcommittee 65E: Devices and integration in enterprise systems, of IEC technical committee 65: Industrial-process measurement, control and automation. It is an International Standard.

This third edition cancels and replaces the second edition published in 2021. This edition constitutes a technical revision.

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

a) added ScanExtended Method.

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

Draft	Report on voting
65E/859/CDV	65E/916/RVC

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

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

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

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

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

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

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

INTRODUCTION

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

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

FIELD DEVICE INTEGRATION (FDI®) –

Part 7: Communication Devices

1 Scope

This part of IEC 62769 specifies the elements implementing communication capabilities called Communication Devices ~~(IEC 62769-5)~~.

The overall FDI®[1] architecture is illustrated in Figure 1. The architectural components that are within the scope of this document have been highlighted in this illustration. The document scope with respect to FDI® Packages is limited to Communication Devices. The Communication Server shown in Figure 1 is an example of a specific Communication Device.

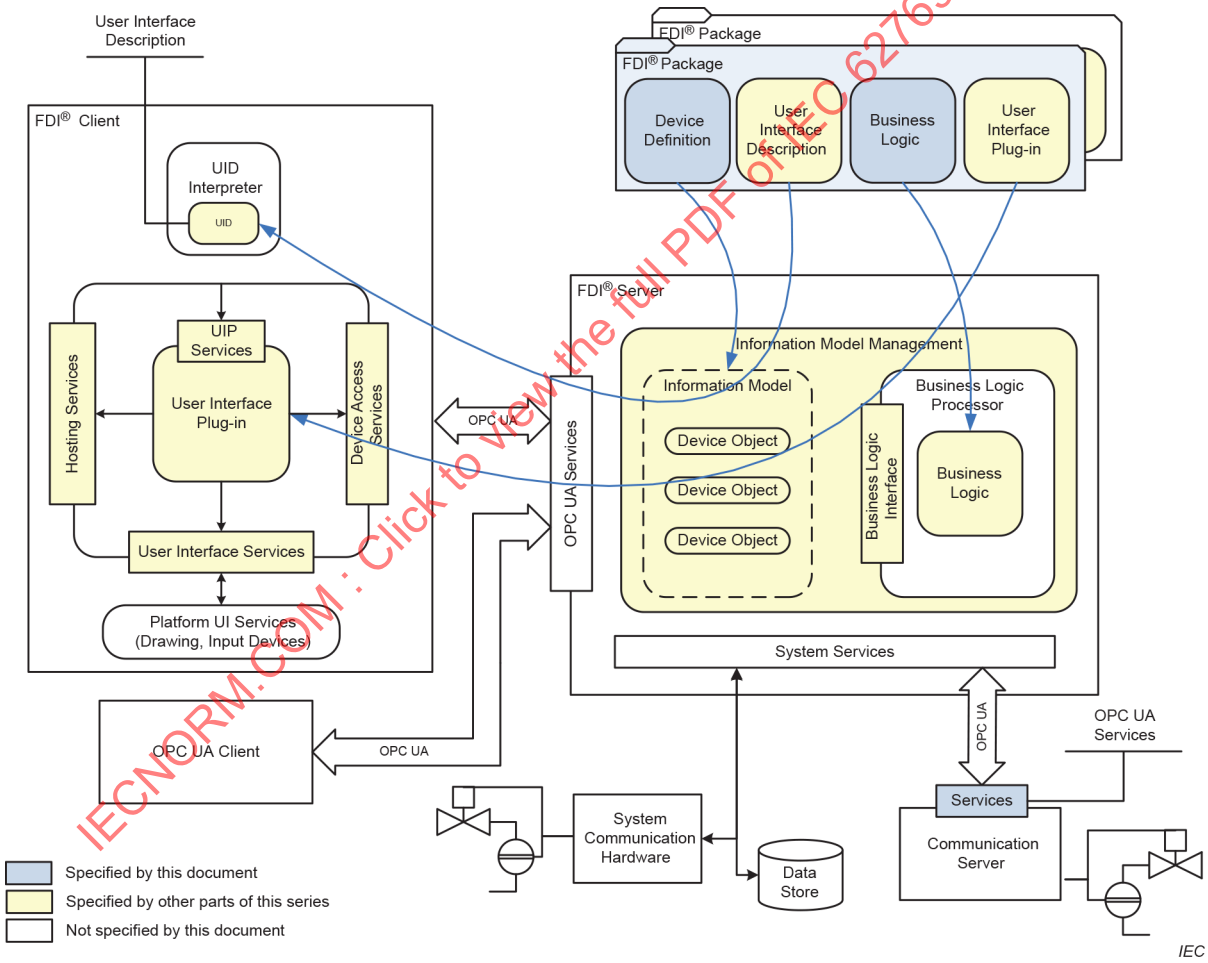


Figure 1 – FDI® architecture diagram

[1] FDI® is a registered trademark of the non-profit organization Fieldbus Foundation, Inc. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the trademark holder or any of its products. Compliance does not require use of the trade name. Use of the trade name requires permission of the trade name holder.

2 Normative references

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

IEC 61804-3, *Devices and integration in enterprise systems − Function blocks (FB) for process control and electronic device description language (EDDL) − Part 3: EDDL syntax and semantics*

IEC 61804-4, *Devices and integration in enterprise systems − Function blocks (FB) for process control and electronic device description language (EDDL) − Part 4: EDD interpretation*

~~IEC 62541 (all parts), *OPC Unified Architecture*~~

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

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

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

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

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

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

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

IEC 62769-3, *Field Device Integration (FDI®) – Part 3: ~~FDI~~ Server*

IEC 62769-4:~~2020~~2023, *Field Device Integration (FDI®) – Part 4: FDI® Packages*

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

3 Terms, definitions, abbreviated terms, acronyms and conventions

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 62769-1, IEC 62769-3, IEC 62541-6 and the following apply.

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

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

3.1.1
Gateway
Communication Device that enables to bridge between different physical networks or different protocols

3.2 Abbreviated terms and acronyms

For the purposes of this document, the abbreviated terms and acronyms given in IEC 62769-1, IEC 62541-6 and the following apply.

HTTP Hypertext Transfer Protocol
IP Internet Protocol
PHY Physical communication hardware
SNMP Simple Network Management Protocol
TCP Transmission Control Protocol
URI Uniform Resource Identifier

3.3 Conventions

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

3.3.1 EDDL syntax

This part of IEC 62769 specifies content for the EDD component that is part of FDI® Communication Packages. The specification content using EDDL syntax uses the font `Courier New`. The EDDL syntax is used for method signature, variable, data structure and component declarations.

3.3.2 Capitalizations

Capitalization of the first letter of words is used in the IEC 62769 series to emphasize an FDI® defined term.

3.3.3 Graphical notation

This document uses the graphical notation defined in IEC 62769-5.

4 ~~General~~ Overview

The abstract term FDI® Communication Device represents an entity implementing communication functions over a network using a specific protocol. The group of FDI® Communication Devices splits into two main groups.

a) The FDI® Communication Server is a dedicated OPC UA Server providing access to one or more field device networks. The FDI® Communication Server is specified in Clause 7.
b) The FDI® Communication Gateway enables to bridge between different physical networks or different protocols. The bridging business logic is implemented in the EDD component that is provided with an FDI® Communication Package. The FDI® Communication Gateway is specified in Clause 8.

NOTE The main differences between a Gateway and a Communication Server are as follows:
In terms of FDI®, the FDI® Communication Server is a dedicated OPC UA Server providing access to one or more field device networks. A Gateway is a Communication Device that enables to bridge between different physical networks or different protocols. The logical representation of a Gateway device within the FDI® Server hosted Information Model enables the FDI® Server to process communication in heterogeneous network topologies.

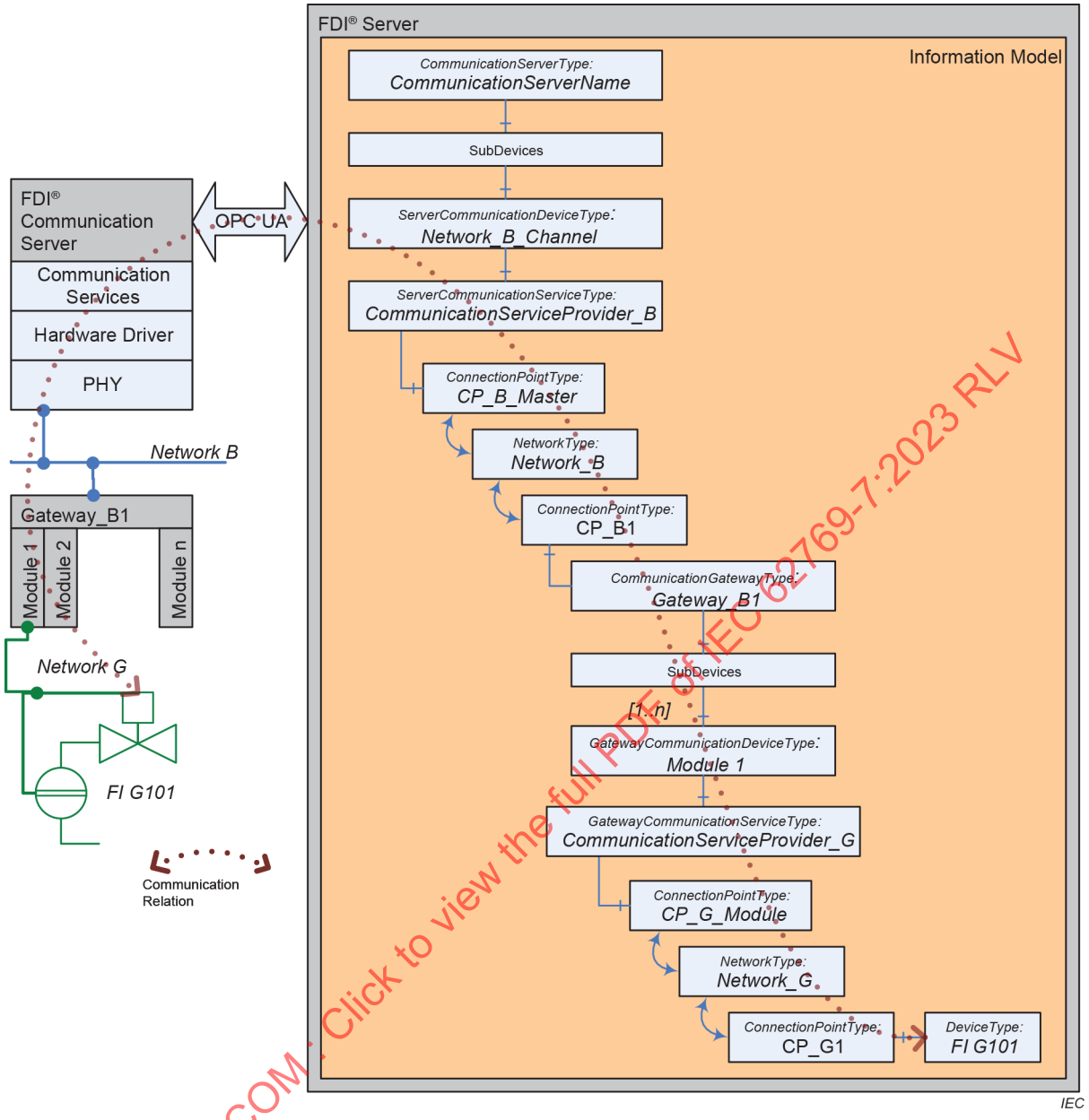


Figure 2 – FDI® communication infrastructure architecture

The FDI® Server hosted Information Model contains a representation of the network topology. (see also IEC 62769-5). The Information Model shown in Figure 2 is an example excerpt to illustrate how the Information Model used elements reflect the actual network topology.

1) The instance of CommunicationServerType (named CommunicationServerName) represents the FDI® Communication Server. The FDI® Communication Server implements physical communication network access (Communication hardware). Clause 7 describes related Information Model specifics, required FDI® Communication Package content and handling of elements therein. (For subdevices, see IEC 62769-5)
2) The instance of ServerCommunicationDeviceType and ServerCommunication-ServiceType (named Network_B_Channel) maps to the FDI® Communication Server implemented communication services. The ServerCommunicationDeviceType is specified in 7.3.3. The ServerCommunicationServiceType is specified in 7.3.4.
3) The instance of CommunicationGatewayType (named Gateway_B1) represents the physical Gateway. Clause 8 describes the related Information Model specifics, the required FDI® Package content and the handling of elements therein.

4) The instance of GatewayCommunicationDeviceType (named Module 1) maps to a physical or logical module enabling communication to the network to which this module is connected. The GatewayCommunicationDeviceType is specified in 8.3.2.3. The related Gateway specifics are described in Clause 8.

5) The instance of GatewayCommunicationServiceType (named CommunicationServiceProvider_G) represents the Gateways' ability to process communication services. The Gateway specific implementation of GatewayCommunicationServiceType is based on Business Logic that enables to run communication services in heterogeneous communication networks.

6) A communication relation (more details are described in Clause 6) between a physical device and the device representation managed by the FDI® Server is always associated to communication service objects that are instances of a GatewayCommunicationServiceType or ServerCommunicationServiceType. The ability of instantiating multiple communication service objects supports protocols enables to operate multiple logical connections between a bus master and a device.

7) The Information Model represents the connections between the physical devices shown on the left side of Figure 2 based on instances of ConnectionPointType NetworkType and the depicted relations. ConnectionPointType and NetworkType are specified in ~~IEC 62769-5~~ IEC 62541-100.

5 FDI® Communication Package

5.1 General

The FDI® Server imports the FDI® Communication Package like any other FDI® Device Package. Clause 5 specifies the FDI® Communication Package details.

5.2 EDD

5.2.1 General rules

The FDI® Communication Package contained EDD is not restricted but bound to a protocol specific profile (see IEC 62769-4:~~2020~~2023, Annex F).

The EDD elements as specified in the protocol specific profile documents (see IEC 62769-4:2023, Annex F), and provided with an FDI® Communication Package shall describe:

a) Parameter and parameter structures. Mandatory protocol specific parameter definitions are found in the protocol specific profile documents (see IEC 62769-4:~~2020~~2023, Annex F). The parameters shall contain any parameter that requires adjustment for proper communication service operation.

b) Physical Layer identification. Protocol specific definitions are found in IEC 62769-4:~~2020~~2023, Annex F.

c) Communication Devices modularity: The modularity information shall be based on using the EDDL constructs COMPONENT (see IEC 61804-3).

   FDI® envisions Communication Device modularity to cope with communication hardware providing multiple physical or logical communication channels to access multiple logical or physical communication networks. Each module element of the whole Communication Device shall be described by a separate EDD element.

d) The COMPONENT definition shall be used to support the system implemented topology configuration. Protocol specific definitions are found in the protocol specific profile documents (see IEC 62769-4:~~2020~~2023, Annex F). The related COMPONENT definitions are described in 5.2.2, 5.2.3, 5.2.4, and 5.2.7.

e) The Business Logic shall contain a method enabled to validate the network (see 5.2.8). The validation function considers the elements only directly connected to the network. The

validation function shall be referred by the EDDL specified CHECK_CONFIGURATION attribute.

f) The Business Logic can contain a method enabled to validate the module configuration (see 5.2.9) or the network configuration (see 5.2.8). The validation function considers the elements only directly connected to the related parent element in the topology. The validation function shall be referred by the EDDL specified CHECK_CONFIGURATION attribute.

g) Connection Point data: The Connection Point (see 5.2.4 and 5.2.6) shall be described through EDDL constructs COMPONENT, COLLECTION and VARIABLE. The COMPONENT definition associates the Connection Point element to the Communication Device. The VARIABLE definitions represent the properties of a specific Connection Point. The COLLECTION represents the Connection Point structure as such. Protocol specific definitions are found in IEC 62769-4:~~2020~~2023, Annex F. Annex A describes a convention for protocol specific annex creation.

h) MENU:

The Menu structure shall follow the Menu conventions for PC based applications according to IEC 61804-4, enabling access to

1) FDI® Communication Device Type (Bus) parameters: These parameters shall be made accessible by means of ""offline_root_menu";

2) Topology Configuration Dialogs shall be made available by means of the menu entry point "topology_configuration".

5.2.2 Device component

Each FDI® Communication Package shall contain an EDD element describing the device.

```
COMPONENT <DeviceComponentId>
{
     LABEL "<Label>";
     CAN_DELETE TRUE;
     CHECK_CONFIGURATION <ValidateModules>;
     CLASSIFICATION NETWORK_COMPONENT;
     COMPONENT_RELATIONS
     {
        <CommunicationDeviceRelationId>
     }
}



COMPONENT_RELATION <CommunicationDeviceRelationId>
{
    LABEL "Relation type description";
    RELATION_TYPE CHILD_COMPONENT;
    ADDRESSING {<AddressVar>}
    COMPONENTS
    {
       <CommunicationDeviceComponentId>
        {
           AUTO_CREATE <autoCreate>;
           REQUIRED_RANGES
             {
               <AddressVar>{ MIN_VALUE <AddrMin>; MAX_VALUE <AddrMax>;}
    }
}
    }
    MINIMUM_NUMBER <minNumber>;
```

```
    MAXIMUM_NUMBER <maxNumber>;
}
```

<DeviceComponentId>: The COMPONENT identifier identifies the component description for the device type.

<Label>: The string value shall contain a string that allows a human user to determine the function of the FDI® Communication Server object.

<ValidateModules>: The Value refers to the METHOD implementing the module topology configuration validation function. (Implementation details are specified in 5.2.9.)

The attribute COMPONENT_RELATIONS allows to describe how modules can be connected. The definition of the COMPONENT_RELATIONS is optional. If used, it shall describe the relations to the CommunicationDevice definitions. The construct enables to perform generic, FDI® Server driven (device) topology configuration. Syntax details are described in IEC 61804-3. The subsequent text describes the semantic use of the COMPONENT_RELATION construct.

<CommunicationDeviceRelationId>: The attribute value identifies the COMPONENT_RELATION definition describing the relation between the device component and the CommunicationDevice component.

<CommunicationDeviceComponentId>: The attribute value ~~has~~ needs to match with a COMPONENT identifier used in a COMPONENT declaration that describes a CommunicationDevice (see 5.2.3).

<autoCreate>: The attribute value describes the number of CommunicationDevice components that can be automatically instantiated with the Device component.

<minNumber>/<maxNumber>/<autoCreate>: The attribute values define the instantiation constraints. The definition of these attributes is optional. The attribute values can contain conditional expressions.

The RELATION_TYPE shall be set to CHILD_COMPONENT.

<AddressVar>: The attribute value is a reference to a VARIABLE declaration. This VARIABLE holds the address value for a CommunicationDevice instance. The definition of this attribute is optional.

<AddrMin>/<AddrMax>: Values define the address value range for a CommunicationDevice instance. The value can, for example, correspond to a physical slot number. Usage of attributes ADDRESSING and REQUIRED_RANGES enables generic configuration routines.

5.2.3 CommunicationDevice component

Each FDI® Communication Package shall contain at least one EDD element describing at least one CommunicationDevice component. A modular communication hardware structure shall be described by multiple CommunicationDevice COMPONENT descriptions:

```
COMPONENT <CommunicationDeviceComponentId>
{
    LABEL "<Label>";
    CAN_DELETE <CanDelete>;
    CLASSIFICATION NETWORK_COMPONENT;
    COMPONENT_RELATIONS
    {
```

```
<CommunicationServiceProviderRelationId>
    }
}
```

```
COMPONENT_RELATION <CommunicationServiceProviderRelationId>
{
   LABEL "Relation between CommunicationDevice and communication
service provider";
   RELATION_TYPE CHILD_COMPONENT;
   ADDRESSING {<AddressVar>}
   COMPONENTS
   {
      <CommunicationServiceProviderId>
      {
         AUTO_CREATE <autoCreate>;
}
   }
   MINIMUM_NUMBER 1;
   MAXIMUM_NUMBER <maxNumber>;
}
```

<CommunicationDeviceComponentId>: The COMPONENT identifier identifies the CommunicationDevice component.

<Label>: The string value shall contain a human readable string that allows a user to easily determine the function of the CommunicationDevice component.

<CanDelete>: Allowed values are TRUE or FALSE. It depends on whether a CommunicationDevice needs explicit configuration or whether the related communication service provider object shall be automatically instantiated with the CommunicationDevice. If the attribute CAN_DELETE is set to FALSE, the CommunicationDevice configuration is static.

The definition of the COMPONENT_RELATIONS is mandatory. It describes the relation to the communication service provider definition. The construct enables the FDI® Server to instantiate communication service provider components according to communication processing demands. (Syntax details are described in IEC 61804-3.) The subsequent text describes the semantic use of the COMPONENT_RELATION construct.

<CommunicationServiceProviderRelationId> The attribute value identifies the COMPONENT_RELATION definition as such.

<CommunicationServiceProviderId>: The attribute value has to match with a COMPONENT identifier used in a COMPONENT declaration that describes a communication service provider (5.2.4).

<autoCreate>: The attribute value describes the number of communication service providers that can be automatically instantiated with the CommunicationDevice component.

The RELATION_TYPE shall be set to CHILD_COMPONENT.

The PROTOCOL attribute shall not be set.

5.2.4 Communication service provider component

Each FDI® Communication Package describing a Communication Device shall contain at least one EDD element describing the communication service provider. The EDD component shall not define any configuration parameter.

```
COMPONENT <CommunicationServiceProviderId>
{
    LABEL "<Label>";
    BYTE_ORDER <byteOrder>;
    CAN_DELETE <CanDelete>;
    CLASSIFICATION NETWORK_COMMUNICATION_SERVICE_PROVIDER;
    COMPONENT_RELATIONS
    {
<CommunicationServiceProvidersConnectionPointRelationId>
    }
}

COMPONENT_RELATION
<CommunicationServiceProvidersConnectionPointRelationId>
{
   LABEL "Relation between communication service provider and
connection point";
   RELATION_TYPE CHILD_COMPONENT;
   ADDRESSING {<AddressVar>}
   COMPONENTS
   {
      < ConnectionPointId>
      {
         AUTO_CREATE 1;
}
   }
   MINIMUM_NUMBER 1;
   MAXIMUM_NUMBER 1;
}
```

<CommunicationServiceProviderId>: The COMPONENT identifier identifies the communication service provider.

<Label>: The string value shall contain a human readable string that allows a user to easily determine the function of the communication service provider object.

<CanDelete>: Allowed values are TRUE or FALSE. It depends on whether a communication service provider can be flexibly instantiated according to the communication processing demands. If the attribute CAN_DELETE is set to FALSE, the number of communication service provider component instantiations is static. The instantiation constraints declared through the attributes AUTO_CREATE, MINIMUM_NUMBER and MAXIMUM_NUMBER correspond to the capabilities of currently supported protocols.

<byteOrder>: The value enables generic integration of n-byte data types (e.g. 4-byte Integer) into the communication message payload. The attribute value describes the byte order and shall be either BIG_ENDIAN or LITTLE_ENDIAN.

The definition of the COMPONENT_RELATIONS is mandatory. It describes the relation to the Connection Point definition. The construct enables to perform generic, FDI® Server driven topology configuration. (Syntax details are described in IEC 61804-3.) The subsequent text describes the semantic use of the COMPONENT_RELATION construct.

The Connection Point shall automatically be instantiated with the communication service provider and there shall be exactly one (1) Connection Point instance connected to the

communication service provider. The instantiation constraints declared through the attributes AUTO_CREATE, MINIMUM_NUMBER and MAXIMUM_NUMBER correspond to the capabilities of currently supported protocols.

<CommunicationServiceProvidersConnectionPointRelationId> The attribute value identifies the COMPONENT_RELATION declaration as such.

<ConnectionPointId>: The attribute value has to match with a COMPONENT identifier used for a COMPONENT declaration that describes a Connection Point (see 5.2.5).

The RELATION_TYPE shall be set to CHILD_COMPONENT.

The PROTOCOL attribute shall not be set.

5.2.5 Connection Point component

Each FDI® Communication Package describing a Communication Device shall contain one EDD element describing one Connection Point for each of the protocols that are supported by the Communication Device:

```
COMPONENT <ConnectionPointId>
{
    LABEL "<Label>";
    CAN_DELETE FALSE;
    CLASSIFICATION NETWORK_CONNECTION_POINT;
    PROTOCOL <ProtocolId>;
    CONNECTION_POINT <ConnectionPointCollectionId>;
}
```

<ConnectionPointId>: The COMPONENT identifier identifies the Connection Point component declaration.

<Label>: The string value shall contain a string that allows a human user to determine the function of the Connection Point component.

<ProtocolID>: The value of this attribute indicates the communication capability which allows the FDI® Server to find other device types that can be connected to the network using the same type of protocol. For standardized protocols, the value is defined by the related field bus organization.

<ConnectionPointCollectionId>: The attribute value is a reference to a COLLECTION declaration that describes the data structure of the Connection Point as described in 5.2.6 .

5.2.6 Connection Point collection

Each EDD describing the Connection Point of a Communication Device shall describe the COLLECTION element that describes the attributes that shall appear in the Information Model representation of the Connection Point. The protocol specific data exposed by the Connection Point identifies the device type and its network address.

```
COLLECTION <ConnectionPointCollectionId>
{
    LABEL "<Label>";
    MEMBERS
    {
        <AddressAttributeName>, <AddressAttributeVariableId>;
        VALID <VALID_VariableId>;
    }
```

```
}
```

<ConnectionPointCollectionId>: The identifier of the COLLECTION is referred by the CONNECTION_POINT attribute value defined in 7.7.3.5.

<Label>: The label identifies the Connection Point in a human readable way.

<AddressAttributeName>/<AddressAttributeVariableId>: The MEMBER section refers to the VARIABLE definitions describing the address attributes implemented by a Connection Point. The content of the MEMBER section is protocol specific.

<VALID>/<VALID_VariableId> is a Collection member referring a Boolean VARIABLE holding the validation status that shall be set by the ValidateNetwork Action (see 5.2.8).

5.2.7 Network component

Each FDI® Communication Package describing a Communication Device shall contain one EDD element describing one Network for each of the protocols that are supported by the Communication Device. The definition supports the network topology engineering:

```
COMPONENT <NetworkComponentId>
{
    LABEL "<Label>";
    CAN_DELETE TRUE;
    CHECK_CONFIGURATION <Validate>;
    CLASSIFICATION NETWORK;
    PROTOCOL <ProtocolId>;
    COMPONENT_RELATIONS
    {
        <NetworksConnectionPointRelationId>
    }
}



COMPONENT_RELATION <NetworksConnectionPointRelationId>
{
   LABEL "Relation between network and connection point";
   RELATION_TYPE CHILD_COMPONENT;
   ADDRESSING {<AddressVar>}
   COMPONENTS
   {
      <ConnectionPointId>
      {
        REQUIRED_RANGES
          {
            <BusAddressVar>{ MIN_VALUE <BusAddrMin>; MAX_VALUE
<BusAddrMax>;}
   }
}
   }
   MINIMUM_NUMBER 1;
   MAXIMUM_NUMBER <maxNumber>;
}
```

<NetworkComponentId>: The COMPONENT identifier identifies the Network component declaration.

<Label>: The string value shall contain a human readable string that allows a user to easily determine the function of the Network component.

<Validate>: The Value refers to the METHOD implementing the network topology configuration validation function (see 5.2.8).

<ProtocolID>: The value of this attribute allows the FDI® Server to find other device types that can be connected to the network using the same type of protocol. For standardized protocols, the value is defined by the related fieldbus organization.

The definition of the COMPONENT_RELATIONS is mandatory. It describes the relation to the Connection Point definition and by that the capabilities of a network. The construct enables to perform generic, FDI® Server driven network topology configuration. Syntax details are described in IEC 61804-3. The subsequent text describes the semantic use of the COMPONENT_RELATION construct.

<NetworksConnectionPointRelationId> The attribute value identifies the COMPONENT_RELATION definition.

<ConnectionPointId>: The attribute value has to match with a COMPONENT identifier used for a COMPONENT declaration that describes a Connection Point (see 5.2.4).

<maxNumber: The attribute value limits the number of Connection Points that can be connected to the network. The attribute values can contain conditional expressions.

The RELATION_TYPE shall be set to CHILD_COMPONENT.

<BusAddressVar>: The attribute value is a reference to a VARIABLE declaration. This VARIABLE holds the network address value for any device that is connected to the network.

<BusAddrMin>/<BusAddrMax>: Values define the network address value range.

5.2.8 ValidateNetwork

The method ValidateNetwork represents the Communication Device implemented Business Logic that validates a current network topology. The ValidateNetwork method handles any necessary dependencies related to bus parameters. The implementation of related EDDL logic is based on the EDDL Builtin function ObjectReference, which enables to analyze a set of child instances (Connection Point instances). The validation logic shall set the <VALID> attribute of the Connection Point instance that has passed the validation.

The implementation of ValidateModules is optional if the module setup is either static or if the configuration rules defined in the COMPONENT construct are sufficient to configure the module setup.

Table 1 shows the ValidateNetwork Action arguments.

Signature

```
ValidateNetwork(
  [out] ~~Integer~~Int32  ServiceError,
  [out] String ErrorMessage);
```

Table 1 – ValidateNetwork Action arguments

Argument	Description
ServiceError	0: OK -1: Failed / the Connection Point that did not pass the validation is indicated by the <VALID> attribute () value set to false. Remark: The argument values correspond to the IEC 61804-3 specified error codes named BI_SUCCESS (value = 0) and BI_ERROR (value = -1). The Action returns the ServiceError result using the "return" statement.
ErrorMessage	If the method returns an empty string (NULL) the Action call succeeded. In case of an error, the Action can return a problem description.

5.2.9 ValidateModules

The method ValidateModules validates the current module setup. The implementation of the related EDDL logic is based on the EDDL Builtin function ObjectReference, which enables to analyze a set of child instances. The implementation of ValidateModules is optional if the module setup is either static or if the configuration rules defined in the COMPONENT construct are sufficient to configure the module setup.

NOTE The decision whether ValidateModules is needed or not is vendor specific.

Table 2 shows the ValidateModules Action Arguments.

Signature

```
ValidateModules(
  [out] ~~Integer~~Int32  serviceError,
  [out] String ErrorMessage);
```

Table 2 – ValidateModules Action arguments

Argument	Description
ServiceError	0: OK -1: Failed / the Connection Point that did not pass the validation is indicated by the <VALID> attribute () value set to false. Remark: The argument values correspond to the IEC 61804-3 specified error codes named BI_SUCCESS (value = 0) and BI_ERROR (value = -1). The Action returns the ServiceError result using the "return" statement.
ErrorMessage	If the Action returns an empty string (NULL), the method call succeeded. In case of an error, the Action can return a problem description.

5.2.10 UIP specifics

The FDI® Communication Package can contain the UIP to support e.g. diagnostics and parameterization.

5.2.11 Deployment

The FDI® Server imports the FDI® Communication Package. The handling of EDD and UIP parts matches with the import procedure performed for the FDI® Package (see IEC 62769-2 and IEC 62769-3).

6 Communication relations

The purpose of a Communication Device and its communication services is to exchange information between the physical device and the device representation managed by the FDI® Server. The information exchange is managed via communication relations, see Figure 3. An established communication relation represents the capability to exchange information between the FDI® Server managed device representation and the physical device. The use of a Communication relation allows abstracting from protocol specifics typically used to manage connections.

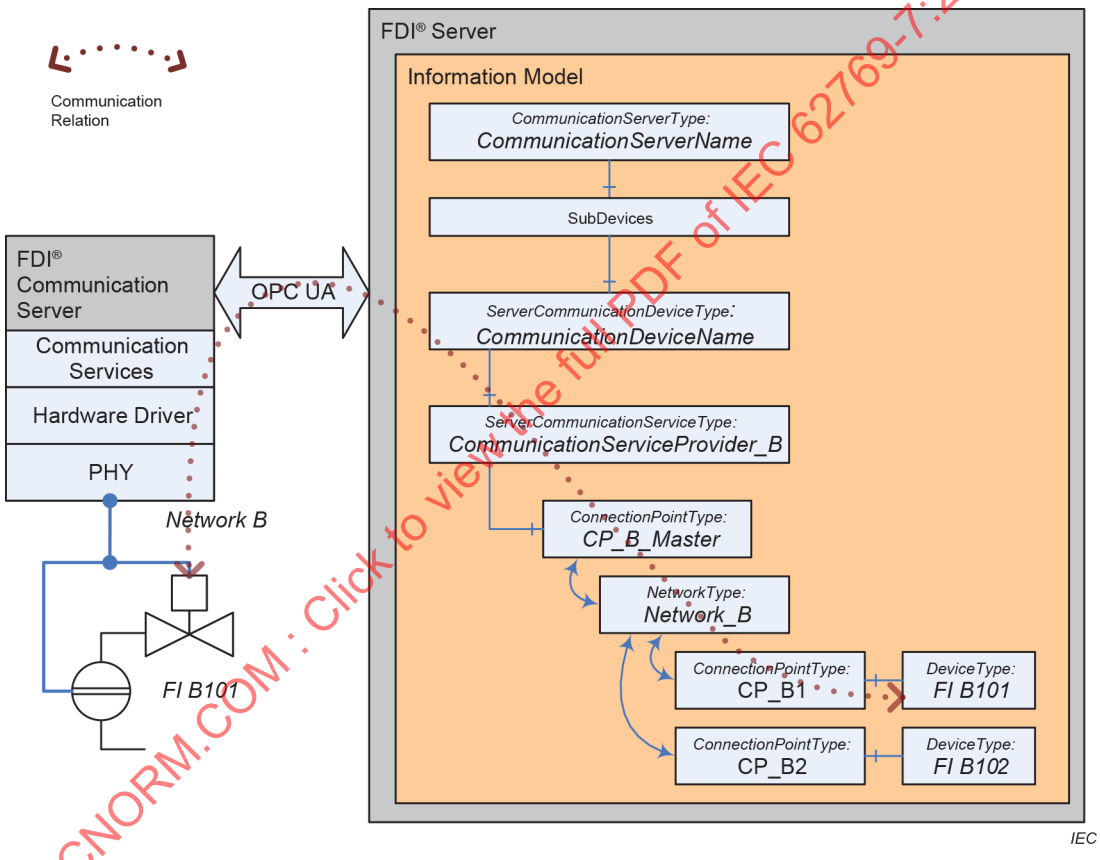


Figure 3 – Communication relation

NOTE 1 The core of information exchange happens between the physical network connected device and the corresponding instance within the Information Model but does not cover the complete device application.

The following state chart describes the general state flow for a single communication relation. The diagram also shows which communication services can be invoked during a "CR Online" state.

The "AbortIndication" shown in Figure 4 can be detected in different protocol specific ways. The one specified for any Communication Device is bound to the serviceErrors returned by the specified communication services. Even the Scan Method can determine a connection loss, when the device for which a communication relation has been activated does not appear in a scan result.

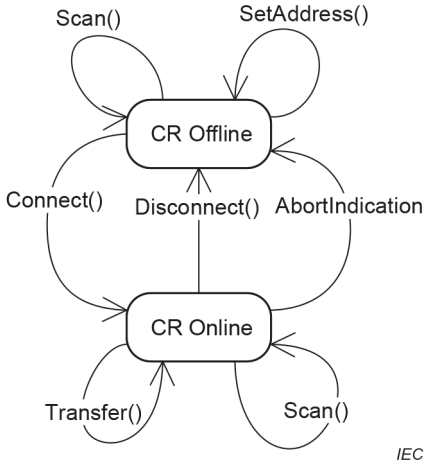


Figure 4 – Communication relation state chart

NOTE 2 The management of communication relations is optional.

7 FDI® Communication Server definition

7.1 General

In terms of FDI®, the FDI® Communication Server is a dedicated OPC UA Server providing access to one or more field device networks. Each FDI® Communication Server is modelled as a Modular Device where each module (also called CommunicationDevice in the sequence) represents the access point to one network.

The Modular Device itself represents the FDI® Communication Server as a whole.

7.2 General characteristics

The FDI® Communication Server implements characteristics for each of its CommunicationDevices specified in 7.3.3. Additionally, an FDI® Communication Server implements the following characteristics:

- The FDI® Server always synchronizes (see 7.5, 7.8.8, and 7.8.11) the FDI® Communication Server hosted Information Model from the FDI® Server hosted Information Model content.
- CommunicationDevices can be statically instantiated or they can be created/deleted by the FDI® Server.
- Communication between the FDI® Server and the FDI® Communication Server is based on OPC UA. OPC UA specifies a wire protocol for its services that can be implemented on arbitrary platforms and runtime environments.
- To avoid race conditions, the FDI® Communication Server only allows one FDI® Server being connected at a time. With this restriction, an FDI® Communication Server can refrain from any synchronization (locking) mechanism. The FDI® specification does not enforce FDI® Communication Servers implementing any interlocking mechanism to manage concurrent access to a single physical network connected device.

7.3 Information Model

7.3.1 General

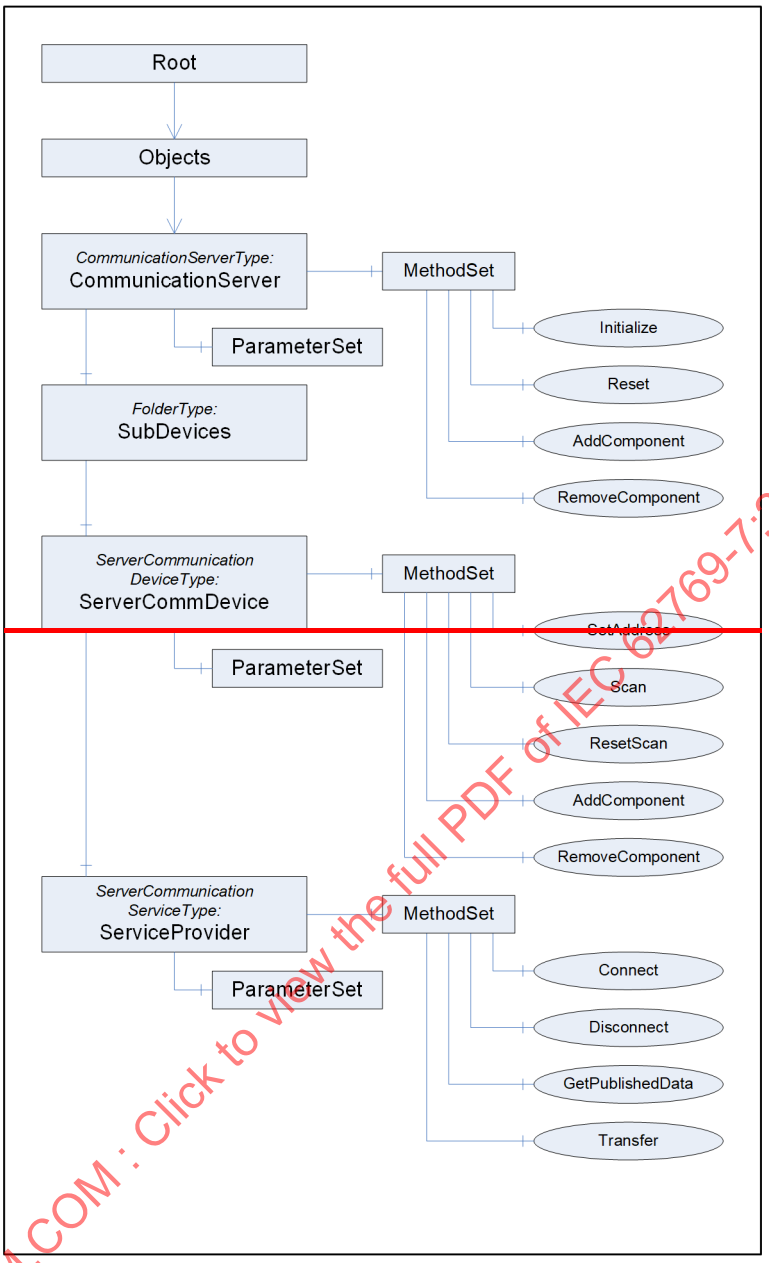
Subclause 7.3 specifies the FDI® Communication Server hosted Information Model.

An FDI® Communication Server is an OPC UA Server that encapsulates communication hardware and provides standardized communication ability. The FDI® Server connects to the FDI® Communication Server as an OPC UA Client and accesses the networks supported by the

FDI® Communication Server via the FDI® Communication Server Information Model. The task of the FDI® Communication Server is to expose this Information Model. The FDI® Communication Server shall not maintain Device Instances or network topology information. All interaction with FDI® devices is done through the FDI® Server and just transferred by the FDI® Communication Server.

For the FDI® Server, an FDI® Communication Server looks like a device that supports FDI® Communication Services and uses OPC UA to communicate. The FDI® Communication Server ~~may~~ can run locally on the same PC as the FDI® Server (loop back adapter) or remote in the field (e.g., embedded into a controller). Like a device, each FDI® Communication Server has an associated FDI® Package. This FDI® Package is used to create Communication Devices in the Information Model of the FDI® Server that represent access to the networks implemented by the FDI® Communication Server.

The Information Model of an FDI® Communication Server is based on the Information Model defined in IEC 62769-5. Figure 5 replicates the Modular Device structure and illustrates how it maps into the overall AddressSpace. The modules represent the communication channels of the FDI® Communication Server. Figure 5 defines the BrowseNames for the nodes.



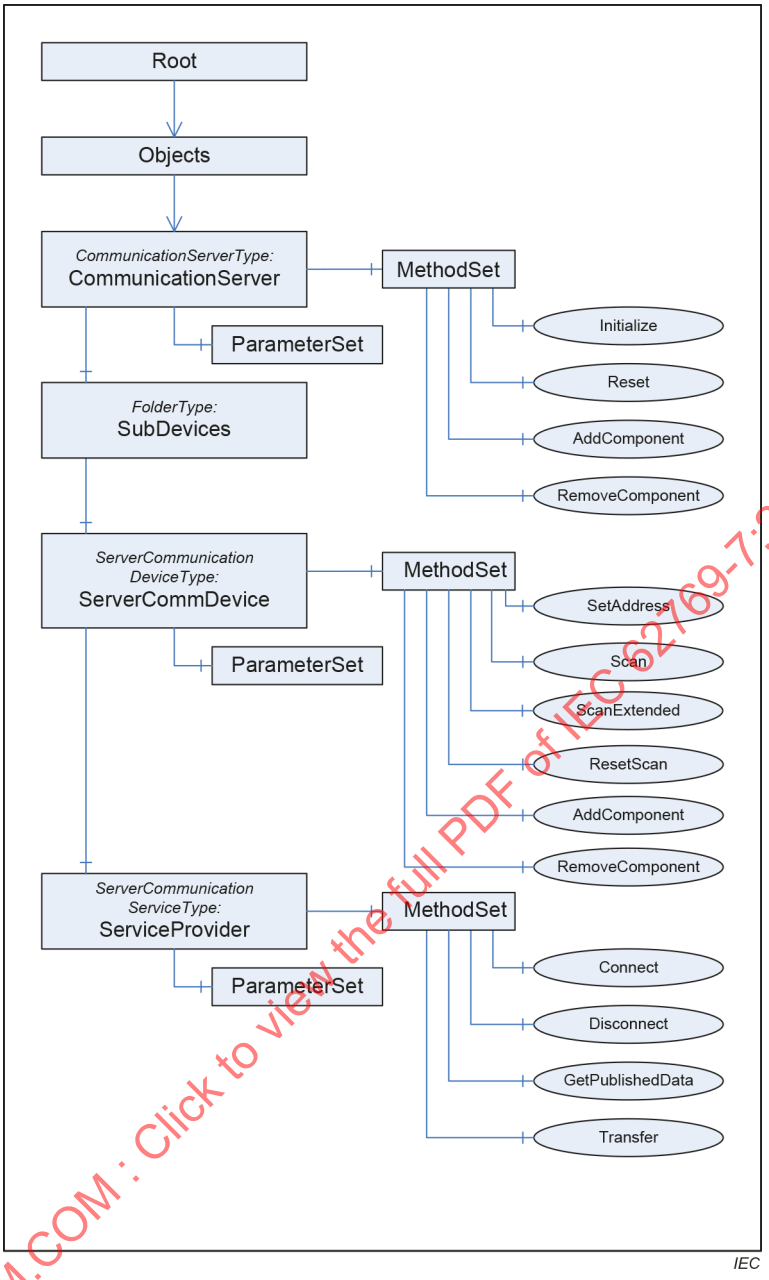


Figure 5 – FDI® Communication Server AddressSpace

The CommunicationServerType (the root of the Modular Device) is a subtype of the DeviceType. The MethodSet contains the methods Initialize, Reset, AddComponent and RemoveComponent. The methods AddComponent and RemoveComponent are optionally present if the FDI® Communication Server supports the dynamic instantiation of elements in the folder SubDevices.

All sub devices are instances of the ServerCommunicationDeviceType defined in 7.3.3. The instances of the ServerCommunicationDeviceType (ServerCommDevice) have a MethodSet that can implement the methods SetAddress, Scan, AddComponent, RemoveComponent. AddComponent and RemoveComponent are optionally present if the FDI® Communication Server supports a variable number of instances of the ServerCommunicationServiceType.

Formal definitions are found in 7.3.2, 7.3.3 and 7.3.4.

7.3.2 CommunicationServerType

7.3.2.1 General

The CommunicationServerType is a subtype of the DeviceType and provides the methods needed to manage the instances ServerCommunicationDeviceType. Figure 6 shows the CommunicationServerType definition that is formally defined in Table 3 and Table 4.

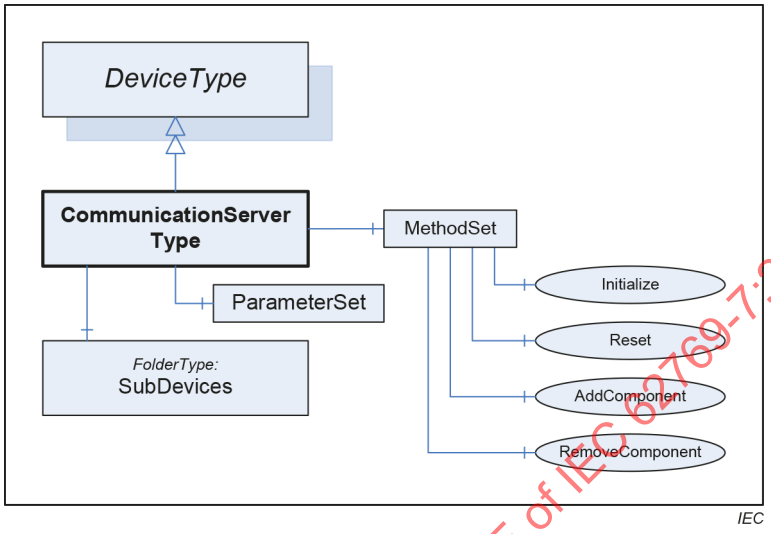


Figure 6 – CommunicationServerType

Table 3 – CommunicationServerType definition

Attribute	Value				
BrowseName	CommunicationServerType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the DeviceType defined in IEC 62541-100.					
HasComponent	Object	MethodSet		BaseObjectType	Mandatory
HasComponent	Object	ParameterSet		BaseObjectType	Optional
HasComponent	Object	SubDevices		FolderType	Mandatory

Table 4 – MethodSet of CommunicationServerType

Attribute	Value				
BrowseName	MethodSet				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
HasComponent	Method	Initialize			Mandatory
HasComponent	Method	Reset			Mandatory
HasComponent	Method	AddComponent			Optional
HasComponent	Method	RemoveComponent			Optional

The CommunicationServerType and each instance of this Type share the same Methods. The NodeId of these Methods will be fixed and defined in this document (see Annex B). FDI®

Communication Server clients therefore do not have to browse for these Methods. They can use the fixed NodeId as the MethodId of the Call Service.

The additional Methods AddComponent and RemoveComponent add the ability to add or remove instances of ServerCommunicationDeviceType according to communication hardware structure. These services are not applicable if the FDI® Communication Server implements a static communication hardware structure.

The SubDevices folder contains instances of ServerCommunicationDeviceType that represent the communication modules.

NOTE The indication for a static communication hardware layout is indicated in the FDI® Package with COMPONENT attribute CAN_DELETE set to FALSE in COMPONENT declarations.

7.3.2.2 Reset Method

Reset is used to reset the communication hardware and related driver software. Any ongoing communication will be stopped immediately. All communication channels enter the status closed.

The Method Reset shall not be present in the FDI® Server hosted Information Model. The FDI® Server shall be able to handle the shut-down procedure automatically according to communication demands.

Typically, the FDI® Communication Server operation includes some hardware and protocol driver handling that can be independent from any modular structure. Because of this possibility, the Reset Method is arranged underneath the CommunicationServerType. For the purpose of reducing the complexity of FDI® Communication Server operation, only one Reset Method has been specified.

The signature of this Method is specified below. Table 5 and Table 6 specify the arguments and AddressSpace representation, respectively.

Signature

```
Reset(
  [out] ~~Integer~~Int32  serviceError);
```

Table 5 – Reset Method arguments

Argument	Description
serviceError	0: OK -1: Failed

Table 6 – Reset Method AddressSpace definition

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

7.3.2.3 Initialize Method

Initialize is used to initialize the communication hardware. The initialization function of the FDI® Communication Server shall use the parameterization data hosted by the ParameterSet that is

contained within the instance of the CommunicationServerType and all instances of ServerCommunicationDeviceType.

In order to enable parameter changes during operation, the Initialize method can be re-invoked. If the FDI® Communication Server needs to reset its communication hardware, it shall automatically restore any communication relation that existed. A modular FDI® Communication Server can flexibly initialize only those ServerCommunicationDeviceType instances for which configuration changes have been detected.

The Method Initialize shall not be present in the FDI® Server hosted Information Model. The FDI® Server shall be able to handle the start procedure automatically according to human driven communication requests.

The FDI® Communication Server operation can include some hardware and protocol driver handling that can be independent from any modular structure. Because of this possibility, the Initialize method is arranged underneath the CommunicationServerType. For the purpose of reducing the complexity of FDI® Communication Server operation, only one Initialize method has been specified.

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

Signature

```
Initialize(
  [out] ~~Integer~~Int32  serviceError)
```

Table 7 – Initialize Method arguments

Argument	Description
serviceError	0: OK -1: Failed

Table 8 – Initialize Method AddressSpace definition

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

7.3.2.4 AddComponent Method

AddComponent shall be used to configure the modular setup of an FDI® Communication Server in case the FDI® Communication Sever has no statically defined communication hardware setup. This method shall be used to add a module (Instance of ServerCommunicationDeviceType).

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

Signature

```
AddComponent(
  [in]    String    ModuleTypeName,
  [in]    String    InstanceName,
  [in]    String    InstanceLabel,
  [out]   NodeId    InstanceNodeId,
  [out] ~~Integer~~Int32     ServiceError);
```

Table 9 – AddComponent Method arguments

Argument	Description
ModuleTypeName	Type of module to be created as defined in the FDI® Package. The module type name shall correspond to one of the COMPONENT identifier definitions (see 5.2.3).
InstanceName	Non-localized name of the module's Device Node of the created element. This name has to be unique within the scope of the FDI® Communication Server's Information Model.
InstanceLabel	Human readable label for the root Node of the created module.
InstanceNodeId	Callee-assigned identifier for the module's Device Node.
ServiceError	0 – OK -1 – E_InvalidType – a module for the specified Type cannot (not anymore) be added -2 – E_DuplicateName – there exists already a module with the same name as specified with the InstanceName argument -3 – E_UnknownType – an unknown ModuleTypeName has been specified -4 – E_LimitExceeded – the total number of modules is exceeded (this might be caused by power constraints or other resource limitations)

Table 10 – AddComponent Method AddressSpace definition

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

7.3.2.5 RemoveComponent Method

RemoveComponent shall be used to remove a module (Instance of ServerCommunicationDeviceType). Implementation of RemoveComponent is optional if the communication hardware setup is static.

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

Signature

```
RemoveComponent(
  [in]  NodeId      ModuleNodeId,
  [out] ~~Integer~~Int32       ServiceError);
```

Table 11 – RemoveComponent Method arguments

Argument	Description
ModuleNodeId	The value is the identification of the existing instance in the Information Model.
ServiceError	0: OK -1: Failed, the specified node does not exist

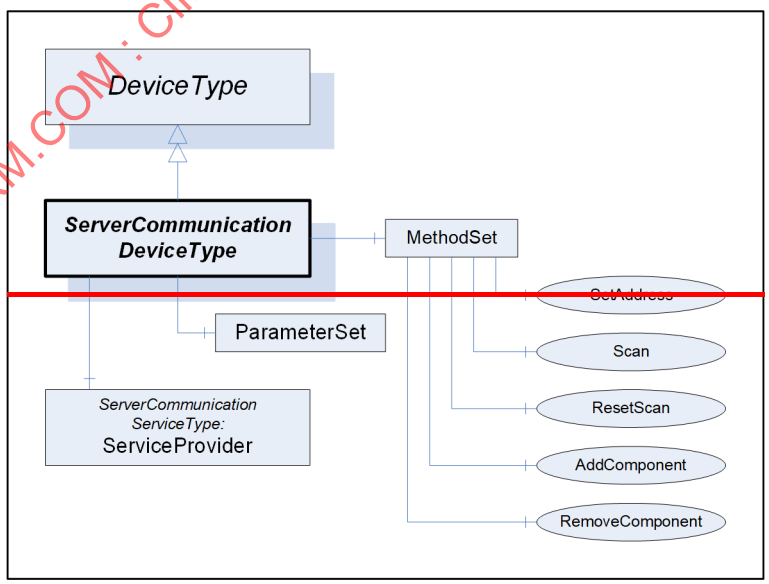
Table 12 – RemoveComponent Method AddressSpace definition

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

7.3.3 ServerCommunicationDeviceType

7.3.3.1 General

The ServerCommunicationDeviceType represents a communication channel for a particular network. The ServerCommunicationDeviceType is a subtype of the DeviceType. The ParameterSet for each instance of a ServerCommunicationDevice will contain Parameters necessary to configure the operation of the network. Figure 7 shows the ServerCommunicationDeviceType definition that is formally defined in Table 13 and Table 14.



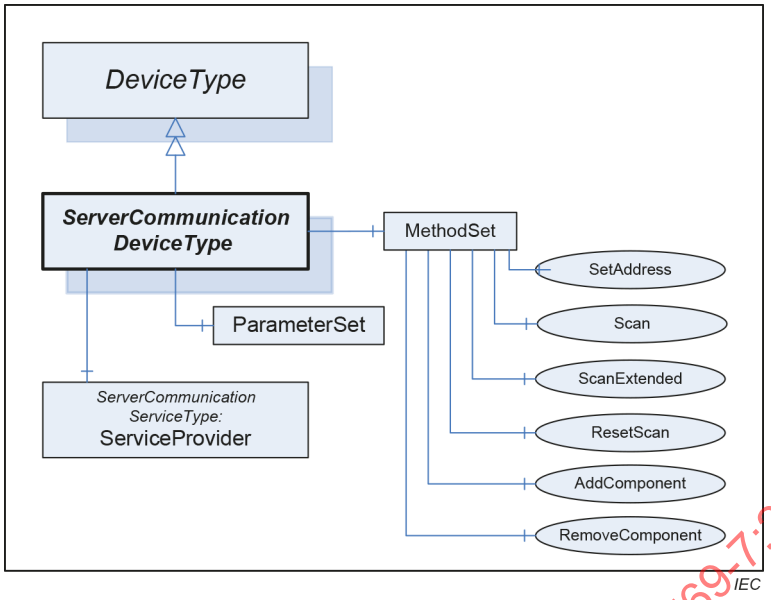


Figure 7 – ServerCommunicationDeviceType

Table 13 – ServerCommunicationDeviceType definition

Attribute	Value				
BrowseName	ServerCommunicationDeviceType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the DeviceType defined in OPC UA Part DI.					
HasComponent	Object	MethodSet		BaseObjectType	Optional
HasComponent	Object	ParameterSet		BaseObjectType	Optional
HasComponent	Object	ServiceProvider		ServerCommunication ServiceType	Mandatory
HasProperty	Variable	ListOfCommunicationProfiles	String[]	PropertyType	Mandatory

The Property ListOfCommunicationProfiles contains a list of communication profiles supported by the ServerCommunicationDevice. Valid strings are defined in IEC 62769-4:~~2020~~2023, Annex F.

Table 14 – MethodSet of ServerCommunicationDeviceType

Attribute	Value				
BrowseName	MethodSet				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
HasComponent	Method	Scan			Optional
HasComponent	Method	ScanExtended			Optional
HasComponent	Method	ResetScan			Optional
HasComponent	Method	SetAddress			Optional
HasComponent	Method	AddComponent			Optional

HasComponent	Method	RemoveComponent			Optional

7.3.3.2 Scan Method

Scan shall be used to start discovering physical network connected devices. The associations between the method Scan and the corresponding physical network connection enables the FDI® Communication Server to access the correct physical network connection. The Scan Method is implemented by the Communication Server runtime module.

The signature of this Method is specified below. Table 15 and Table 16 specify the arguments and AddressSpace representation, respectively.

NOTE 1 Communication Servers can run the network scan in a background task so the invocation of the function Scan will return cached network scan results.

NOTE 2 In case the SCAN takes very long, the FDI® Communication Server might return an empty TopologyScanResult and the ServiceError 1 identifying that the scan is still running.

Signature

```
Scan(
  [out] XmlElement      TopologyScanResult,
  [out] ~~Integer~~Int32          ServiceError)
```

Table 15 – Scan Method arguments

Argument	Description
TopologyScanResult	The argument value is an XML formatted string representing a list of physical network connected devices. Each of the physical network connected devices is represented by a data structure matching with a Connection Point node. Connection Point attributes are protocol specific. The corresponding topologyScanResult schema is specified in IEC 62769-4:~~2020~~2023, Annex F. Return an empty string for TopologyScanResult in case of any error.
ServiceError	0: OK / scan completed 1: OK / get complete scan result by calling Scan again -1: Failed / not initialized -2: Failed / not connected to a network -3: Failed / no device found, the topologyScanResult is empty

Table 16 – Scan Method AddressSpace definition

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

7.3.3.3 ScanExtended Method

ScanExtended shall be used to start discovering physical network connected devices, if the FDI® Server and the FDI® Communication Server support at least FDI® Technology Version 1.3. This method performs the same functionality as the Scan Method. It returns more detailed information about the devices found on the network. The ScanExtended Method is implemented by the Communication Server runtime module. ScanExtended is only available if the communication profile requires the method.

The signature of this Method is specified below. Table 17 and Table 18 specify the arguments and AddressSpace representation, respectively.

NOTE 1 Communication Servers can run the network scan in a background task so the invocation of the function Scan will return cached network scan results.

NOTE 2 In case the SCAN takes very long, the FDI® Communication Server might return an empty TopologyScanResultExtended and the ServiceError 1 identifying that the scan is still running.

Signature

```
Scan(
  [out] XmlElement      TopologyScanResultExtended,
  [out] Int32           ServiceError)
```

Table 17 – Scan Method arguments

Argument	Description
TopologyScanResultExtended	The argument value is an XML formatted string representing a list of physical network connected devices. Each of the physical network connected devices is represented by a data structure matching with a Connection Point node. Connection Point attributes are protocol specific. The corresponding topologyScanResultExtended schema is specified in IEC 62769-4:2023, Annex E. Return an empty string for TopologyScanResultExtended in case of any error.
ServiceError	0: OK / scan completed 1: OK / get complete scan result by calling Scan again -1: Failed / not initialized -2: Failed / not connected to a network -3: Failed / no device found, the topologyScanResult is empty

Table 18 – Scan Method AddressSpace definition

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

7.3.3.4 ResetScan Method

ResetScan shall be used to reset the internal cache of scan results. It will also cancel a running scan in case the FDI® Communication Server scan mechanism supports this.

The signature of this Method is specified below. Table 19 and Table 20 specify the arguments and AddressSpace representation, respectively.

Signature

```
ResetScan(
  [out] ~~Integer~~Int32      ServiceError)
```

Table 19 – ResetScan Method arguments

Argument	Description

ServiceError	0: OK / scan reset -1: Failed / not initialized -2: Failed / not connected to a network

Table 20 – ResetScan Method AddressSpace definition

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

7.3.3.5 SetAddress Method

SetAddress shall be used to change the network address (communication address) of a device. The Communication Device shall ensure unique network address values. If the argument value of newAddress is already assigned to a physical network connected device the Communication Device shall return the argument serviceError value "-4: Failed / duplicate Address error".

It depends on the protocol whether the address assignment service shall work even when a communication relation is already established.

The signature of this Method is specified below. The arguments for SetAdress Method are described in Table 21.

Signature

```
SetAddress(
   [in]                    <OldAddress>,
   [in]                    <NewAddress>,
   [out]  Int32            ServiceError);
```

Table 21 – SetAddress Method arguments

Argument	Description
<OldAddress>	This is a placeholder for a 1..n protocol specific arguments representing the existing protocol specific network address of the physical network connected device. Values that represent a network address are specified in protocol specific profile documents (see IEC 62769-4:~~2020~~2023 Annex F).
<NewAddress>	This is a placeholder for 1..n protocol specific arguments representing the new protocol specific network address that shall be assigned to a physical network connected device. Values that represent a network address arguments are specified in IEC 62769-4:~~2020~~2023, Annex F.
ServiceError	0: OK / execution finished successfully Other values that represent protocol specific ServiceErrors ~~are~~ might be specified in protocol specific profile documents (see IEC 62769-4:~~2020~~2023, Annex F).

7.3.4 ServerCommunicationServiceType

7.3.4.1 General

Communication services provide the means to communicate with a Device or to e.g. execute a Scan on some Network. Communication services are represented through Methods in the Information Model (see IEC 62769-5).

The formal definition of ServerCommunicationServiceType is found in Figure 8, Table 22 and Table 23.

The NodeId of these Methods will be fixed and defined in this document (see Annex B). FDI® Clients therefore do not have to browse for these Methods. They can use the fixed NodeId as the MethodId of the Call Service.

Communication methods including their NodeIds are uniquely defined in this document. FDI® Clients can use the Methods directly (without browsing). The OPC UA Call Service shall be used as follows:

- the MethodId argument shall contain the fixed NodeId of the Method;
- the ObjectId argument shall contain the NodeId of the MethodSet.

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

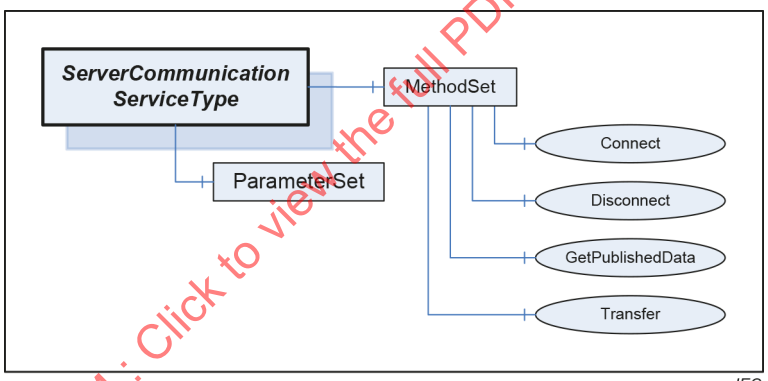


Figure 8 – ServerCommunicationServiceType

Table 22 – ServerCommunicationServiceType definition

Attribute	Value				
BrowseName	ServerCommunicationServiceType				
IsAbstract					
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the DeviceType defined in OPC UA Part DI.					
HasComponent	Object	MethodSet		BaseObjectType	Mandatory
HasComponent	Object	ParameterSet		BaseObjectType	Optional

Table 23 – MethodSet of ServerCommunicationServiceType

Attribute	Value
BrowseName	MethodSet

IsAbstract					
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
HasComponent	Method	Connect			Optional
HasComponent	Method	Disconnect			Optional
HasComponent	Method	Transfer			Mandatory
HasComponent	Method	GetPublishedData			Optional

7.3.4.2 Connect Method

Connect shall be used to establish a communication relation to a device that is physically connected to the network. Establishing the communication relation ~~may~~ can imply checks of identification data that are part of the addressData with data inside the physical device. The Communication Device performs this DeviceType match verification according to a corresponding network protocol standard. Related details are specified in IEC 62769-4:~~2020~~2023, Annex F.

The devices address is contained in the Connection Point of the corresponding Device Instance within the Information Model (Device Connection Point). The communication relation between the Information Model associated device application and the physical device is further on identified by the communication relation identifier. Details about how to manage the status of a communication relation is described in Clause 6. ~~Connection Point descriptions may be layered as described in Annex A.~~

NOTE 1 As the NodeId is a unique identifier within the Information Model scope, the NodeId of the Device Connection Point can be a unique identifier for any communication relation in the scope of a Communication Device.

NOTE 2 The term communication relation is introduced describing the status of an infrastructure that enables data exchange between information model hosted data and a physical device. If the communication relation is established data exchange is possible.

The signature of this Method is specified below. Table 24 specifies the arguments.

Signature

```
Connect(
   [in]   ByteString        CommunicationRelationId,
   [in]                     <AddressData>,
   [out]                    <DeviceInformation>,
   [out]  Int32             ServiceError);
```


Table 24 – Connect Method arguments

Argument	Description
CommunicationRelationId	This is a client generated id that is used to uniquely identify this connection. This could be an index (e.g., a NodeId) that the client (= FDI® Server) needs to identify entries in its topology.
<AddressData>	This is a placeholder for a protocol specific argument list that is used for the address and optional device identification data (details described in IEC 62769-4: ~~2020~~2023, Annex F.
<DeviceInformation>	This is a placeholder for a protocol specific argument list in which the connect result data are stored.
ServiceError	0: OK / execution finished, connection established successfully Other values that represent protocol specific ServiceErrors ~~are~~ might be specified in protocol specific profile documents (see IEC 62769-4:~~2020~~2023, Annex F)

7.3.4.3 Disconnect method

Disconnect shall be used to terminate a communication relation to a Device.

The signature of this Method is specified below. Attributes of the Disconnect method are specified in Table 25. Disconnect is a synchronous method call.

Signature

```
Disconnect(
   [in]   ByteString        CommunicationRelationId,
   [out]  Int32             ServiceError);
```

Table 25 – Disconnect Method arguments

Argument	Description
CommunicationRelationId	Same ID as used in method Connect specified in 7.3.4.2.
ServiceError	1: OK / disconnect finished successfully Other values that represent protocol specific ServiceErrors ~~are~~ might be specified in protocol specific profile documents (see IEC 62769-4:~~2020~~2023, Annex F)

7.3.4.4 Transfer method

Transfer shall be used to perform information exchange with a Device.

The signature of this Method is specified below. All arguments are specified in Table 26.

Signature

```
Transfer(
   [in]   ByteString              CommunicationRelationId,
   [in]                                <SendData>,
   [out]                               <ReceiveData>,
   [out]  Int32                        ServiceError);
```

Table 26 – Transfer Method arguments

Argument	Description
CommunicationRelationId	See 7.3.4.2.
<SendData>	This is a placeholder for a protocol specific list of arguments as described in IEC 62769-4:~~2020~~2023, Annex F. The argument values represent the protocol specific communication service request that is sent to the device.
<ReceiveData>	This is a placeholder for a protocol specific list of arguments as described in IEC 62769-4:~~2020~~2023, Annex F. The argument values represent the protocol specific communication service response that is received from the device.
ServiceError	0: OK / execution finished, ReceivedData contains the result Other values that represent ServiceErrors are specified in IEC 62769-4:~~2020~~2023, Annex F.

7.3.4.5 GetPublishedData Method

The FDI® Server sends GetPublishedData requests to the FDI® Communication Server to receive data that is submitted by unsolicited data messages. The argument SendData contained

data prepares the exchange of "unsolicited" data messages from the device. The content of SendData is protocol specific. The FDI® Communication Server queues GetPublishedData requests in a queue associated with the Communication Relation defined through the argument CommunicationRelationId. The argument PublishId identifies the related queue entry. Each time the FDI® Communication Server receives unsolicited data messages, it saves the received data in association with the existing queue entry that has been created for the GetPublishedData. Depending on the underlying network technology (performance), the method GetPublishedData can immediately return with data coming from an "unsolicited" data message.

Subsequent pulling of data that is submitted by unsolicited data messages works through the same method GetPublishedData. In this case, the argument SendData is empty. The argument PublishId matches with the value that has been provided with the initial call GetPublishedData that has established the transmission of exchange of "unsolicited" data messages.

In order to stop the device sending the "unsolicited" data, the method GetPublishedData shall be used again but the argument SendData contained data terminates the exchange of "unsolicited" data messages from the device. Table 27 shows the GetPublishedData Method arguments.

Signature

```
GetPublishedData(
    [in]   ByteString          CommunicationRelationId,
    [in]                       <SendData>,
    [out]                      <ReceiveData>,
    [out]  DateTime            TimeStamp
    [in]   UInt32              PublishId,
    [out]  Int32               ServiceError);
```

Table 27 – GetPublishedData Method arguments

Argument	Description
CommunicationRelationId	See 7.3.4.2.
<SendData>	This is a placeholder for a protocol specific list of arguments as described in IEC 62769-4:~~2020~~2023, Annex F. The argument values control the exchange of unsolicited messages.
<ReceiveData>	This is a placeholder for a protocol specific list of arguments as described in IEC 62769-4:~~2020~~2023, Annex F. The argument values convey data that comes from unsolicited messages.
TimeStamp	Time, when the data was published by the device
PublishId	The number identifies an established subscription that conveys data that comes from unsolicited messages.
ServiceError	0: OK / execution finished Other values that represent protocol specific ServiceErrors ~~are~~ might be specified in protocol specific profile documents (see IEC 62769-4:~~2020~~2023, Annex F)

7.4 OPC UA Server Profile for FDI® Communication Server

Profiles are named groupings of ConformanceUnits as defined in ~~IEC 62541-6 and~~ IEC 62541-7. The term Facet in the title of a Profile indicates that this Profile is expected to be part of another larger Profile or concerns a specific aspect of OPC UA. Profiles with the term Facet in their title are expected to be combined with other Profiles to define the complete functionality of an OPC UA Server or Client. The minimum required OPC UA Server Profile is the "Micro Embedded Device Server Profile".

The following table specifies the facet for an OPC UA Server that acts as an FDI® Communication Server. Table 28 describes Conformance Units included in this facet.

Table 28 – FDI®CommunicationServer_Facet definition

Conformance Unit	Description	Optional/ Mandatory
FDI® Comunication Server Information Model	Support at least one instance of CommunicationServerType.	M

7.5 Mapping the FDI® Server Information Model to the FDI® Communication Server IM

7.5.1 General

The representation of an FDI® Communication Server in the AddressSpace of an FDI® Server is almost identical to the AddressSpace that exists in the FDI® Communication Server. This refers in particular to the Modular Device hierarchy and the Parameters of all Devices. However, the Nodes in the FDI® Server are built from the device description imported via the FDI® Communication Package.

7.5.2 Information Model differences

Because of their different tasks, however, there are a few differences and a set of synchronization rules. The Information Model example shown in Figure 9 depicts the commonalities and the differences between the Information Models hosted by the FDI® Communication Server and the FDI® Server. In general, the FDI® Communication Server - hosted Information Model is a subset of the FDI® Server hosted Information Model. The Device Instances in the FDI® Server and the FDI® Communication Server adhere to the same type definitions. Thus, browse names of common Information Model elements shall have the same browse name.

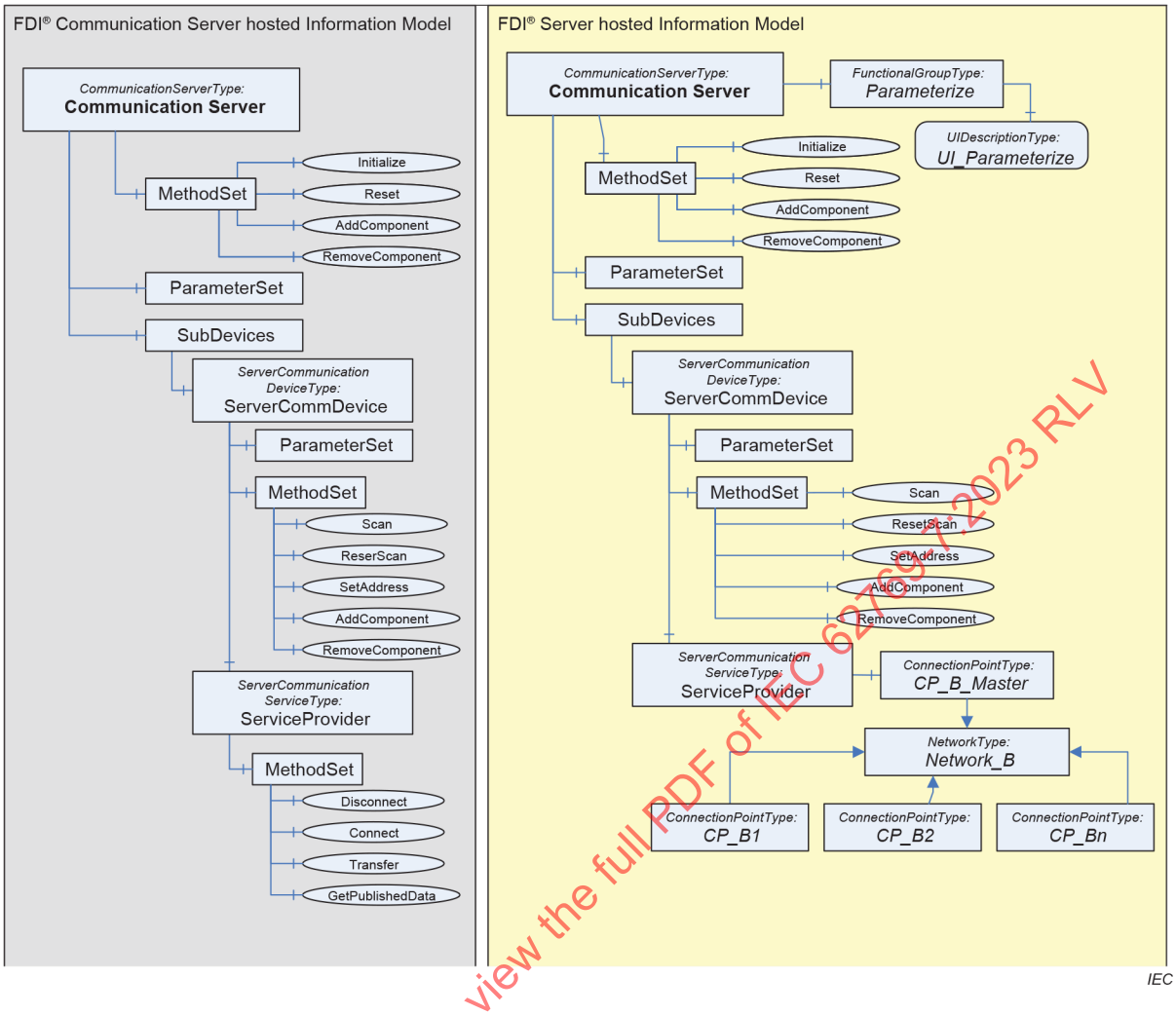


Figure 9 – Information Model differences (example)

The list of differences in the Information Model is as follows:

- The FDI® Server supports online and offline versions of the Modular Device; the FDI® Communication Server supports just an online version. The online version of the FDI® Server represents the version in the FDI® Communication Server, i.e., if Parameter Values are read or written to the online model of the FDI® Server, these operations are passed through to the FDI® Communication Server. This happens both for public and for private Parameters.

  NOTE The key is a match between the browse names present in the FDI® Server hosted Information Model and the FDI® Communication Server hosted Information Model. This allows generic synchronization of both information models.

- UIPs, UIDs, Actions and Functional Groups exist only in the FDI® Server.
- Modules in the FDI® Server hosted Information Model have a Connection Point component that is used to connect this module to a network when creating the Device Topology. The FDI® Communication Server hosted Information Model does not show Connection Point elements. The Device Topology is managed by the FDI® Server only (see IEC 62769-5).
- The FDI® Server can represent the ServiceProvider without exposing the MethodSet in order to prevent an FDI® Client from invoking communication services.

The mapping of the Module Management functionality is as follows:

- AddComponent and RemoveComponent are exposed in the FDI® Communication Server Information Model via the FDI® Communication Server Object (the root of the Modular

Device). They exist only if there are modules to be configured, i.e. they will not be available if the Communication Server does not support modular communication hardware configuration. AddComponent and RemoveComponent replace the generic Node Management service defined by OPC UA.

The FDI® Server handles module topology related configuration based on the Node Management Service Set (see IEC 62769-3 and IEC 61804-4). On any module configuration related activity, the FDI® Server first calls the ValidateModules Action. The EDD Action can run through various states and even perform user dialogs (see description about Actions in IEC 62769-2 and IEC 62769-5). The EDD Methods can maintain (private) information that is global to the Modular Device. The EDD Action can access the module that shall be created.

7.6 Installer

The Installer for the FDI® Communication Server executable is optional. Since the FDI® specification does not prescribe the implementation platform for FDI® Communication Server executables, the FDI® Communication Server executable can be also preinstalled on dedicated hardware.

The installer used for the FDI® Communication Server executable shall be separated from the FDI® Package. Importing the FDI® package is a separate procedure (see 7.7.1).

7.7 FDI® Communication Package

7.7.1 General

The FDI® Server imports the FDI® Communication Package like any other FDI® Package. Subclause 7.7 specifies the FDI® Communication Package details.

With respect to the EDD element of the Package, FDI® differentiates between a simple (lightweight) Communication Server (see 7.7.2) and a regular (multi-channel) Communication Server (see 7.7.3).

7.7.2 EDD for lightweight Communication Server

A lightweight Communication Server provides access to a single field device network. It shall provide all configuration capabilities in its main EDD, not in the sub-modules used to expose the connection points. This allows FDI® Hosts not supporting modular devices to parameterize an FDI® Communication Server using the standard FDI® user interface mechanisms.

The EDD describing a "lightweight" Communication Server shall follow the IEC 61804-3 specified profile for the Communication Server. But it shall not use the following EDDL syntax constructs:

- COMPONENT,
- COMPONENT_RELATION,
- COMPONENT_FOLDER,
- COMPONENT_REFERENCE,
- EDDL Builtin function ObjectReference.

7.7.3 EDD for multi-channel Communication Server

7.7.3.1 General

The required content for an FDI® Communication Package EDD element describing an FDI® Communication Device is specified in Clause 5. Specific EDD element content for an FDI® Communication Server is described in 7.7.3.

The rules defined in 5.2.2 apply.

The PROTOCOL attribute shall not be set.

The COMPONENT declaration shall have an additional attribute PRODUCT_URI. The attribute value holds a string describing the FDI® Communication Server product URI that enables the FDI® Server to identify the FDI® Communication Server based on the OPC UA Discovery service (see IEC 61804-3). The attribute value corresponds to the RegisterServer argument RegisteredServer:serverUri. The product URI shall contain the company name and the product name.

EXAMPLE: PRODUCT_URI "urn:Company:ProductName".

7.7.3.2 CommunicationDevice component

The rules defined in 5.2.3 apply.

7.7.3.3 Communication Service component

The rules defined in 5.2.4 apply.

7.7.3.4 Connection Point component

The rules defined in 5.2.5 apply.

7.7.3.5 Connection Point collection

The rules defined in 5.2.6 apply.

7.7.4 COMMANDs in EDDs for FDI® Communication Servers

An EDD for an FDI® Communication Server shall follow the communication profile CS defined in IEC 61804-3. As the synchronization between the VARIABLEs of the EDD represented as parameters in the FDI® Servers information model and the actual configuration of the FDI® Communication Server (represented as parameters in the information model of the FDI® Communication Server) is done by the FDI® Server, there is no need to define COMMANDs in the EDD in order to access the data of the FDI® Communication Server.

However, the communication profile CS allows the definition of COMMANDs in an EDD. This allows an EDD developer to explicitly trigger communication to the FDI® Communication Server. This ~~may~~ can be needed in complex EDD methods that require to directly write to the FDI® Communication Server or directly read the actual value before proceeding with the method execution.

The COMMAND does not require addressing information (like NUMBER, SLOT, INDEX (see IEC 61804-3) since the addressing is done via the BrowseName of the FDI® Communication Server and the EDD identifier of the VARIABLE. Instead of the RESPONSE of a READ COMMAND shall contain the VARIABLEs to be read and the REQUEST of a WRITE COMMAND shall contain all the VARIABLEs to be written.

If such a COMMAND is called (by ReadCommand or WriteCommand Builtins in the EDD), the FDI® Server shall read respectively write all the VARIABLEs defined in the COMMAND.

Since COMMANDs also group access to VARIABLEs, a FDI® Server shall always handle VARIABLEs defined in a COMMAND as a unit. For example, when a WRITE COMMAND contains the two VARIABLEs VarA and VarB and the FDI® Server needs to write VarA because it was changed by the user, the FDI® Server also needs to write VarB in the same OPC UA service call, although VarB might not have been changed. Same applies for reading VARIABLEs.

OPC UA READ and WRITE service calls return different status codes (see IEC 62541-4). ~~First, they return one service result code per call. Then, they return one operation level result code per parameter to be read or written. Since an EDD COMMAND only supports one RESPONSE_CODE per call, the following mapping applies.~~

~~The first status code that is not GOOD shall be used as RESPONSE_CODE with the following order:~~

~~1) investigate the service result code, then~~

~~2) investigate the operation level result codes in the order the VARIABLEs appear in the COMMAND definition.~~

First, per call one service result code and second per parameter to be read or written one operation level result code. Since an EDD COMMAND only supports one RESPONSE_CODE per call, the following mapping applies: The first status code that is not GOOD shall be used as RESPONSE_CODE with the following order: First investigate the service result code, then the operation level result codes in the order the VARIABLEs appear in the COMMAND definition. If all status codes are GOOD, the RESPONSE_CODE is GOOD.

7.7.5 Documentation

The FDI® Communication Package shall provide documentation describing:

a) the software installation related environment requirements and procedures,

b) the OPC UA Server configuration if needed.

7.8 Handling and behaviour

7.8.1 General

Subclause 7.8 defines the FDI® Communication Server handling and behaviour rules along the life cycle beginning with the deployment, start up, bus commissioning, until the communication services processing. The diagram (see Figure 10) shows the FDI® Server maintained FDI® Communication Server status.

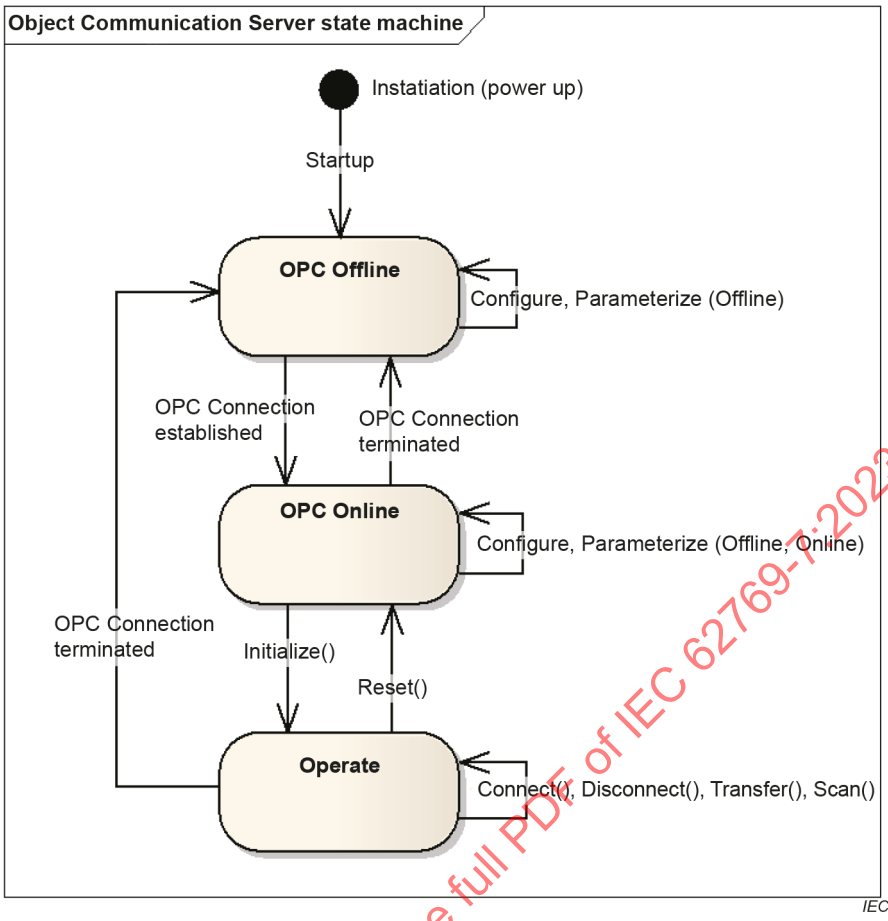


Figure 10 – FDI® Communication Server state machine

7.8.2 Deployment

The FDI® Server imports the FDI® Communication Package. The handling of EDD and UIP parts matches with the import procedure performed for the FDI® Package (see IEC 62769-2 and IEC 62769-3). The FDI® Communication Package represents the Communication Server Type.

The installation procedure described in the following is optional. (An embedded FDI® Communication Server need not provide an installation procedure.)

The FDI® Communication Server Installer (see 7.6) is a separate element. The installation procedure is started manually.

Depending on the operating systems, the execution of installation programs could require administration rights.

7.8.3 Server configuration

The FDI® Communication Server shall implement the means enabling the OPC UA Server specific configuration setting the link to the Discovery Server and the name of the FDI® Communication Server. According to 7.8.4, the FDI® Communication Server needs to know the address information about the Discovery Server. This document does not prescribe the way of how to do this.

The bootstrap process needed to establish the connection between an OPC UA Client and an OPC UA Server requires some administration work. A simple plug and play does not seem possible.

7.8.4 Start up

The following definitions about the Server discovery mechanism refer to the definitions found in IEC 62541-4.

The FDI® Communication Server executable shall be started according to one of the following described ways:

a) The FDI® Communication Server executable is loaded by means of the configured operating system function. If the FDI® Communication Server is installed on a hardware separated from the FDI® Server, the FDI® Communication Server executable shall be loaded by means of the configured operating system function (auto-start).

b) The FDI® Server invokes the FDI® Communication Server executable process. The related functions are specific to the operating system and the system vendor implementations.

The starting FDI® Communication Server process shall register itself at a Discovery Server using the service RegisterServer. This enables OPC UA Clients to obtain information about the connected FDI® Communication Server including the application description, existing endpoints and security information. The related OPC UA Services are FindServers and GetEndPoints. The Discovery Server is a process running outside the FDI® Communication Server.

After starting up, the FDI® Communication Server has the status "OPC Offline".

7.8.5 Shutdown

The following definitions about the Server discovery mechanism refer to the definitions found in IEC 62541-4.

The shutdown of the FDI® Communication Server process shall unregister itself at the Discovery Server using the service RegisterServer by setting argument isOnline value false.

7.8.6 Watchdog

The following definitions about the Server discovery mechanism refer to the definitions found in IEC 62541-4.

The FDI® Communication Server shall periodically use the service RegisterServer to state its ability to receive a connection from the FDI® Server. The frequency is 10 min. The FDI® Communication Server can envision a VARIABLE for configuration purposes.

7.8.7 Establish the OPC UA connection

The FDI® Server connects as an OPC UA Client to the FDI® Communication Server according to ~~the IEC 62541 series~~ IEC 62541-4.

The FDI® Server (OPC UA Client) establishes a secure connection to the FDI® Communication Server (OPC UA Server) using the SecureChannel service set defined in IEC 62541-4. The functional principles are defined in IEC TR 62541-1.

The communication between the FDI® Server (OPC UA client) and the FDI® Communication Server (OPC UA server) shall be based on the OPC UA TCP transport protocol with the OPC UA Binary Encoding and OPC UA Secure Conversation.

After successfully establishing the OPC UA connection, the FDI® Communication Server enters the status "OPC Online".

7.8.8 Instantiate the Communication Server

The creation of a CommunicationServer instance in the FDI® Server hosted Information Model works the same way as the instantiation of a Device.

7.8.9 Configure the communication hardware

As described in 7.3.1, the FDI® Communication Server can support the configuration of modular communication hardware. The modular communication hardware configuration shall be performed via the services AddComponent and RemoveComponent (7.3.2.4, 7.3.2.5).

If the Action ValidateModules (see 5.2.9) is implemented, the FDI® Server shall invoke this method. If the Action result is OK, then the FDI® Server shall perform the synchronization using the sub device browse name matching and the invocation of the FDI® Communication Server hosted module management services.

If the FDI® Communication Server supports modular communication hardware configuration, the correct communication hardware configuration is a prerequisite for successful initialization as described in 7.8.12.

7.8.10 Configure the Network

The COMPONENT declaration defined in 5.2.2, 5.2.3, 5.2.4, and 5.2.7 enables a description based approach for the FDI® Server to configure the network connections. The FDI® Communication Server's ValidateNetwork Action (see 5.2.8) can be added to support the network topology validation function as described in 5.2.8. The network topology is only present in the FDI® Server hosted Information Model (see 7.5).

7.8.11 Parameterize

The FDI® Communication Server can require proper bus parameter adjustment prior to any communication service processing. The FDI® Communication Package contained user dialogs (UIP or UID) enable interactive bus parameter adjustment. The FDI® Communication Package can contain additional Business Logic for bus parameterization purposes. The editing of FDI® Communication Server parameters changes the content of the FDI® Server hosted Information Model. The FDI® Server shall perform synchronization using the parameters' browse ~~names~~ name matching. The FDI® Server copies the modified values from the FDI® Server hosted Information Model to the FDI® Communication Server hosted Information Model.

A simple FDI® Communication Server shall provide all its configuration capabilities in its main EDD, not in the sub-modules used to expose the connection points. This allows FDI® Hosts not supporting modular devices to parameterize an FDI® Communication Server using the standard FDI® user interface mechanisms.

NOTE The FDI® Server can change Parameter values in arbitrary order.

7.8.12 Initialize

On invocation of the method Initialize (see 7.3.2.3), the FDI® Communication Server shall use the current parameter settings and communication hardware configuration for communication hardware initialization purposes.

After successful initialization, the FDI® Communication Server enters the status "Operate".

7.8.13 Create the communication service object

Prior to running any data exchange, at least one instance of type ServerCommunicationServiceType has to be present or created. One instance of ServerCommunicationServiceType shall be always present. Further instances of

ServerCommunicationServiceType can be created if the instance of ServerCommunicationDeviceType implements the method AddComponent.

7.8.14 Communication relation

The definitions of Clause 6 apply. The FDI® Communication Server specifics are defined in the subsequent text.

The state chart in Figure 11 describes the state flow of a single communication relation with added status changes that are related to OPC UA specifics. Beside the specific aspects, the definitions in Clause 6 apply as well.

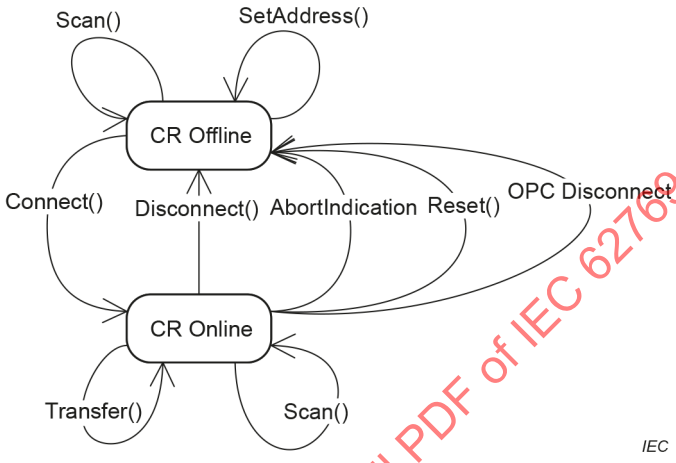


Figure 11 – Communication relation state chart

On invocation of the method Reset or the OPC Connection termination (OPC Connection loss), the FDI® Communication Server shall terminate all communication relations.

The FDI® Communication Server shall reject any parameterization or configuration change attempt in the status "Operate".

The FDI® Communication Server shall reject any communication relation related operation if the FDI® Communication Server status is different than "Operate".

If the Communication Server supports multiple instances of ServerCommunicationServiceType, these instances need to share information about existing communication relations.

7.8.15 Connect

Prior to running any information exchange related communication, the FDI® Communication Server requires establishing a communication relation between the device application and the physical network connected device. This happens through invocation of the method Connect.

The FDI® Communication Server shall be able to manage multiple communication relations. After successful execution of the method Connect, the corresponding communication relation enters the status "CR Online".

Because of the direct association of method Connect to a single communication service provider instance, the Communication Device knows the corresponding physical network connection.

7.8.16 Disconnect

Invocation of the method Disconnect terminates a communication relation, which inhibits further information exchange related communication with the physical network connected device.

After execution of the method Disconnect the corresponding communication relation enters the status "CR Offline". The communication relation becomes invalid.

7.8.17 Abort indication

Depending on protocol specifics, the FDI® Communication Server can detect communication aborts. Such communication abort indications are returned as communication service results during processing of the methods Transfer or Scan. After the FDI® Communication Server has returned an Abort Indication, the current communication relation enters the status "CR Offline". The communication relation becomes invalid.

7.8.18 Scan

The topology scan function can be invoked independently from an existing communication relation. Scan service details are specified in 7.3.3.2.

7.8.19 SetAddress

It depends on the protocol whether the address assignment service shall work even when a communication relation is already established.

8 FDI® Communication Gateway definition

8.1 General

A Gateway is a Communication Device that enables to bridge between different physical networks or different protocols. The logical representation of a Gateway device within the FDI® Server hosted Information Model enables the FDI® Server to process communication in heterogeneous network topologies.

8.2 Information Model

8.2.1 General

The Information Model of a Gateway is based on the Information Model defined in IEC 62769-5. Figure 12 replicates the Modular Device structure and its integration in the overall FDI® Server hosted Information Model.

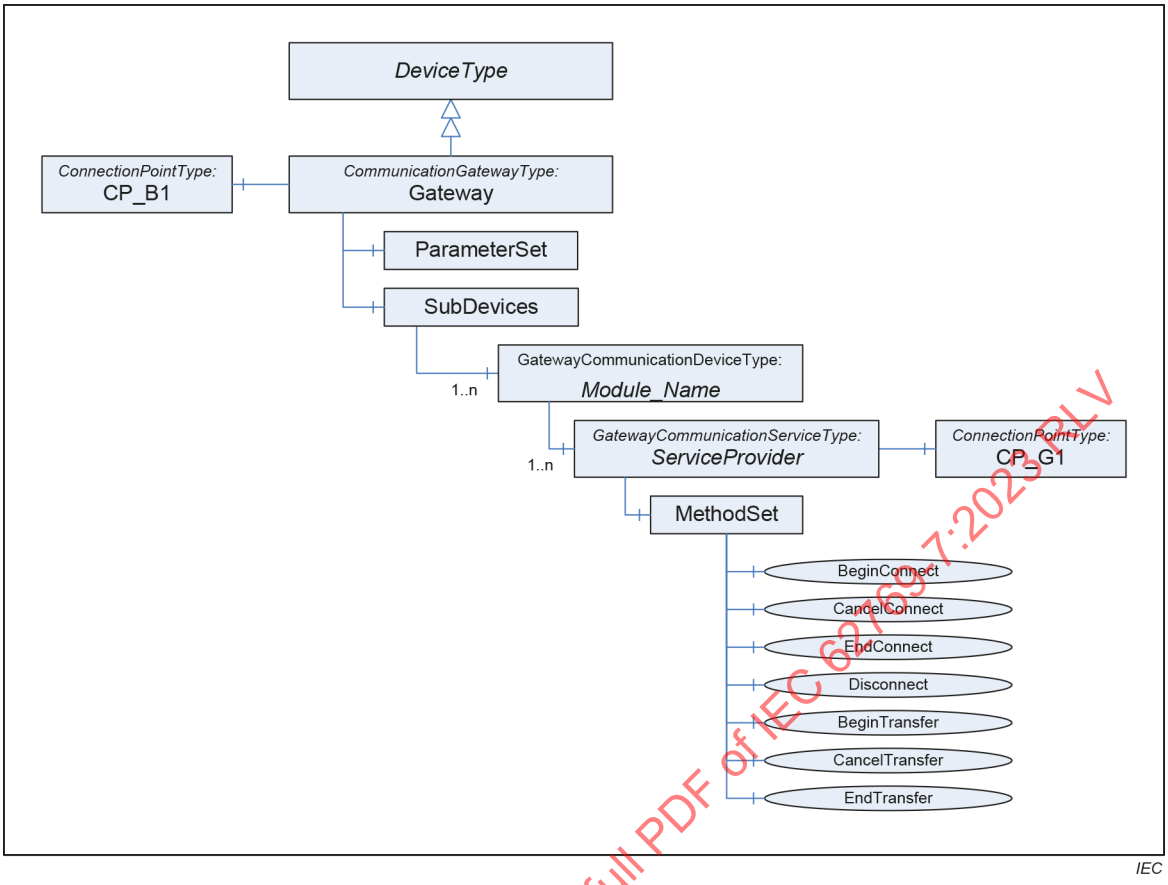


Figure 12 – Gateway information model

The Gateway is connected to the Network (see IEC 62769-5) through an instance of a ConnectionPointType (CP_B1). CP_B1 represents the FDI® Server assigned object (see IEC 62769-5) identification (name).

The Gateway is an instance of DeviceType. The optionally available ParameterSet (see IEC 62769-5) shall contain all device parameters that parameterize the communication interface used for Gateway initiated communication service requests.

The elements underneath SubDevices represent the physical or logical access to the network media of the Gateway. The attribute Module_Name represents the FDI® Server assigned object (see IEC 62769-5) identification (browse name).

The Gateway's communication service processing capabilities are accessible through multiple communication service provider instances created from GatewayCommunicationServiceType. The Business Logic behind the service methods implements the protocol translation function that is associated with the communication service interface.

NOTE Compared to the FDI® CommunicationServer, the Gateway does not support the transport of unsolicited messages, see 7.3.4.5.

8.2.2 CommunicationGatewayType

The CommunicationGatewayType is a subtype of the DeviceType. Figure 13 shows the CommunicationGatewayType definition. It is formally defined in Table 29.

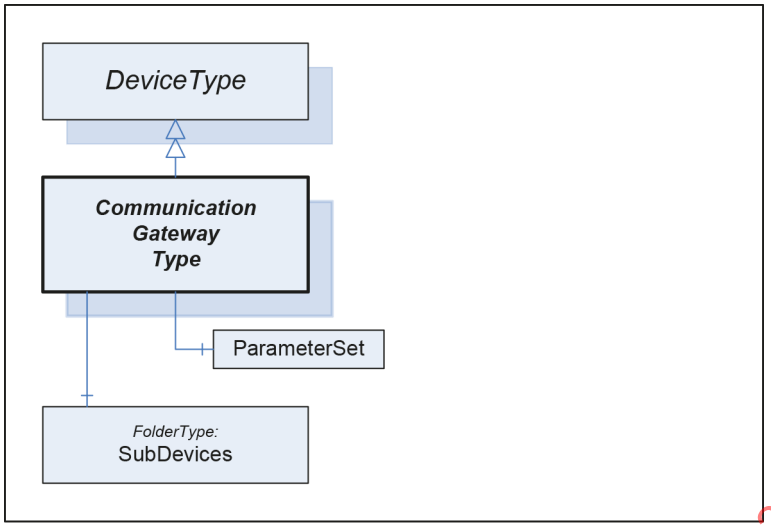


Figure 13 – CommunicationGatewayType

Table 29 – CommunicationGatewayType definition

Attribute	Value				
BrowseName	CommunicationGatewayType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the DeviceType defined in IEC 62541-100.					
HasComponent	Object	SubDevices		FolderType	Mandatory
HasComponent	Object	ParameterSet			Optional

The module management needed to support a configurable communication hardware structure is based on the COMPONENT definitions for the entire CommunicationGatewayType. The COMPONENT definitions provide sufficient information to run generic module setup configuration.

NOTE The indication for a static communication hardware layout is present with the COMPONENT attribute CAN_DELETE set to FALSE for all COMPONENT declarations related to the sub devices of the entire device.

8.2.3 GatewayCommunicationDeviceType

8.2.3.1 General

The GatewayCommunicationDeviceType represents a communication module or channel connected to a particular network. The GatewayCommunicationDeviceType is a subtype of the DeviceType. The ParameterSet for each GatewayCommunicationDeviceType will contain Parameters necessary to configure the operation of the network. The protocol specific, mandatory bus parameters are specified in IEC 62769-4:~~2020~~2023, Annex F. Figure 14 shows the GatewayCommunicationDeviceType definition that is formally defined in Table 30 and Table 31.

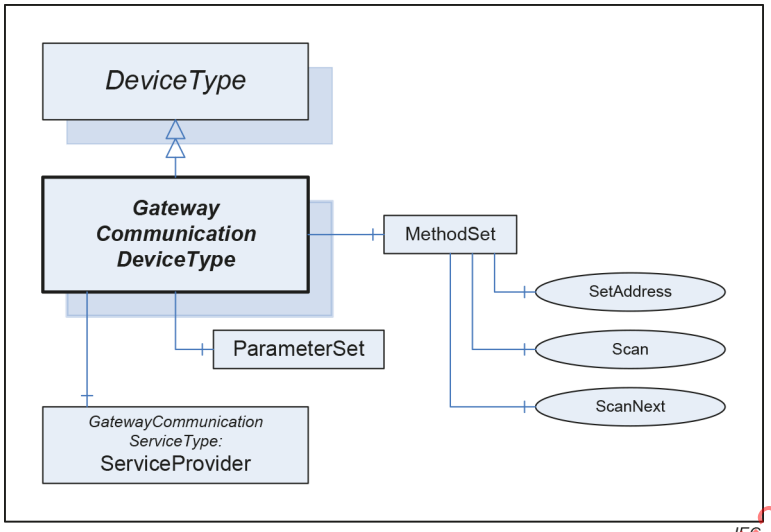


Figure 14 – GatewayCommunicationDeviceType

Table 30 – GatewayCommunicationDeviceType definition

Attribute	Value				
BrowseName	GatewayCommunicationDeviceType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the DeviceType defined in OPC UA Part DI.					
HasComponent	Object	MethodSet		BaseObjectType	~~Optional~~ Mandatory
HasComponent	Object	ParameterSet		BaseObjectType	Optional
HasComponent	Object	ServiceProvider		Gateway Communication ServiceType	Mandatory

Table 31 – MethodSet of GatewayCommunicationDeviceType

Attribute	Value				
BrowseName	MethodSet				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
HasComponent	Method	Scan			Optional
HasComponent	Method	ScanNext			Optional
HasComponent	Method	SetAddress			Optional

8.2.3.2 Scan Method

Scan shall be used to start discovering physical network connected devices. The associations between the method Scan and the corresponding physical network connection enable the FDI® Communication Server to access the correct physical network connection. The Scan Method is implemented by an EDDL based method.

Because EDDL logic is not designed to handle XML documents the Scan service signature deviates from the definition given in 7.3.3.2. The scan service implementation returns the discovered devices using the LIST construct containing the COLLECTION instances. (Details are specified in 8.2.3.4.) The COLLECTION instances represent the Connection Point instances.

The signature of this Method is specified below. Table 32 and Table 33 specify the arguments and AddressSpace representation, respectively.

Signature

```
Scan([out] ~~Integer~~Int32 ServiceError)
```

Table 32 – Scan Method arguments

Argument	Description
ServiceError	0: OK 1: OK / get complete scan result by calling ScanNext -1: Failed / not initialized -2: Failed / not connected to a network -3: Failed / no device found the topologyScanResult is empty

Table 33 – Scan Method AddressSpace definition

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

8.2.3.3 ScanNext Method

ScanNext shall be used to continue discovering physical network connected devices. The associations between the method Scan and the corresponding physical network connection enable the FDI® Communication Server to access the correct physical network connection. The Scan Method is implemented by an EDDL based method.

Because EDDL logic is not designed to handle XML documents, the Scan service signature deviates from the definition given in 7.3.3.2. The scan service implementation returns the discovered devices using the LIST construct containing COLLECTION instances. (Details are specified in 8.2.3.4.) The COLLECTION instances represent the Connection Point instances.

The signature of this Method is specified below. Table 34 and Table 35 specify the arguments and AddressSpace representation, respectively.

Signature

```
ScanNext([out] IntegerInt32 ServiceError)
```

Table 34 – ScanNext Method arguments

Argument	Description
ServiceError	0: OK 1: OK / get complete scan result by recalling ScanNext -1: Failed / not initialized -2: Failed / not connected to a network -3: Failed / no device found the topologyScanResult is empty

Table 35 – ScanNext Method AddressSpace definition

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

8.2.3.4 Scan List

Each EDD describing the Connection Point of a Gateway shall describe the LIST element that holds the list of discovered devices after the successful execution of the Scan service.

```
LIST <Id>
{
    TYPE <ListEntry>;
}
```

<ID>: The identifier shall match with the SCAN _LIST attribute value described in 8.3.2.3.

<ListEntry>: The attribute value shall refer to the COLLECTION definition that describes the Connection Point as defined in 8.3.2.5.

8.2.3.5 SetAddress Method

For definitions, see 7.3.3.5.

8.2.4 GatewayCommunicationServiceType

8.2.4.1 General

Communication services provide the means to communicate with a Device or to e.g. execute a Scan on some Network. Communication services are represented through EDDL based methods (business logic) provided with the FDI® Device Package contained EDD.

The implementation of all communication services except the Disconnect service follows an asynchronous pattern. This implies that each communication service is split into three methods. This allows the FDI® Server to execute communication services in parallel as well as cancel operations that last too long.

Begin<name>: This method starts the execution of the service. The method returns either the execution state of <name> or the result if it is immediately present.

End<name>: This method checks if the result of the service is already available that was started using a preceding Begin<name> call. Like Begin<name> this method returns either the execution state or the result if available.

Cancel<name>: This method cancels a started service execution.

A service identification number (ServiceId) enables the Communication Gateway to keep track of the relation between the method calls that belong to a single communication service process. If the Communication Gateway supports multiple instances of GatewayCommunicationServiceType these instances do share information about currently used ServiceIds. Thus, a communication service has to be executed on a single instance of GatewayCommunicationServiceType.

To reduce unnecessary poll cycles, both methods Begin<name> End<name> return a delay time value (DelayForNextCall). The method caller shall delay the invocation of the method End<name> according to the returned argument value.

Figure 15 shows the GatewayCommunicationServiceType definition that is formally defined in Table 36 and Table 37.

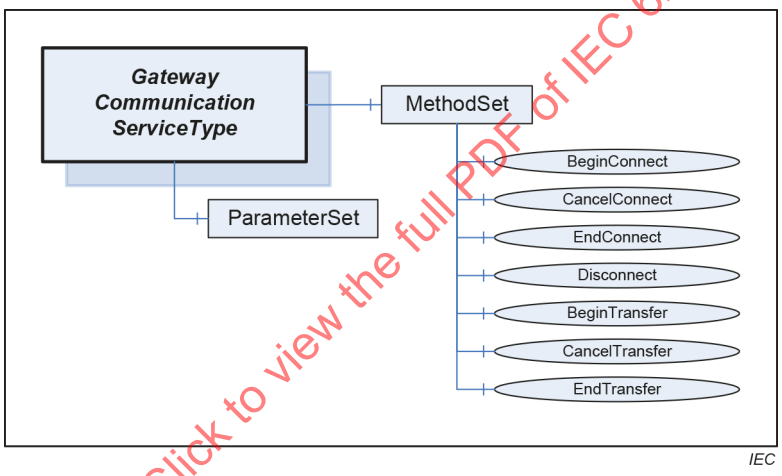


Figure 15 – GatewayCommunicationServiceType

Table 36 – GatewayCommunicationServiceType definition

Attribute	Value				
BrowseName	GatewayCommunicationServiceType				
IsAbstract					
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the DeviceType defined in OPC UA Part DI.					
HasComponent	Object	MethodSet		BaseObjectType	Mandatory
HasComponent	Object	ParameterSet		BaseObjectType	Optional

Table 37 – MethodSet of GatewayCommunicationServiceType

Attribute	Value				
BrowseName	MethodSet				
IsAbstract					
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
HasComponent	Method	BeginConnect			Optional
HasComponent	Method	CancelConnect			Optional
HasComponent	Method	EndConnect			Optional
HasComponent	Method	Disconnect			Optional
HasComponent	Method	BeginTransfer			Mandatory
HasComponent	Method	CancelTransfer			Mandatory
HasComponent	Method	EndTransfer			Mandatory

8.2.4.2 Connect service

The Connect service shall be used to establish a communication relation to a device that is physically connected to the Network. Establishing the communication relation ~~may~~ can imply checks of identification data that are part of the addressData with data inside the physical device. The service performs this DeviceType match verification according to a corresponding network protocol standard. Related details are specified in IEC 62769-4:~~2020~~2023, Annex F.

The devices address is contained in the ConnectionPoint of the corresponding Device Instance within the Information Model (Device Connection Point). The communication relation between the Information Model associated device application and the physical device is further on identified by the communication relation identifier. Details about how to manage the status of a communication relation are described in Clause 6.

NOTE 1 As the NodeId is a unique identifier within the Information Model scope, the NodeId of the Device Connection Point can be a unique identifier for any communication relation in the scope of a Communication Device.

NOTE 2 The term communication relation is introduced to describe the status of an infrastructure that enables data exchange between the information model hosted data and a physical device. If the communication relation is established, data exchange is possible.

The signatures of the Connect service methods are specified below. Table 38 specifies the arguments.

Signature

```
BeginConnect(
   [in]   ByteString       CommunicationRelationId,
   [in]                    <AddressData>,
   [out]                   <DeviceInformation>,
   [in]   UInt32           ServiceID,
   [out]  UInt32           DelayForNextCall,
   [out]  Int32            ServiceError);

EndConnect(
   [in]   ByteString       CommunicationRelationId,
   [out]                   <DeviceInformation>,
   [in]   UInt32           ServiceID,
   [out]  Uint32           DelayForNextCall,
   [out]  Int32            ServiceError);

CancelConnect(
   [in]   ByteString       CommunicationRelationId,
   [in]   Uint32           ServiceID,
   [out]  Int32            ServiceError);
```

Table 38 – Connect Method arguments

Argument	Description
CommunicationRelationId	This is a client generated id that is used to uniquely identify this connection. This could be an index (e.g., a NodeId) that the client (= FDI® Server) needs to identify entries in its topology.
<AddressData>	This is a placeholder for a protocol specific argument list that is used for the address and optional device identification data (details are described in IEC 62769-4:~~2020~~2023, Annex F.
<DeviceInformation>	A protocol specific argument list in which the connect routine stores the resulting data.
ServiceId	The service transaction code establishes the relation between the service request and the corresponding response.
DelayForNextCall	The value specifies a delay time in ms, the caller shall wait before the next invocation of EndConnect.
ServiceError	0: OK / function started asynchronously, result has to be polled with EndConnect 1: OK / execution finished, connection established successfully -1: Connect Failed / cancelled by caller -2: Call Failed / unknown service ID -3: Connect Failed / device not found -4: Connect Failed / invalid device address -5: Connect Failed / invalid device identification

8.2.4.3 Disconnect method

Specified in 7.3.4.3.

8.2.4.4 Transfer service

Transfer shall be used to perform information exchange with a Device.

The signatures of the Transfer service methods are specified below. All arguments are specified in Table 39.

Signature

```
BeginTransfer(
   [in]   ByteString          CommunicationRelationId,
   [in]                             <SendData>,
   [out]                            <ReceiveData>,
   [in]   UInt32              ServiceId,
   [out]  UInt32              DelayForNextCall,
   [out]  Int32                     ServiceError);

EndTransfer(
   [in]   ByteString          CommunicationRelationId,
   [out]                            <ReceiveData>,
   [in]   UInt32              ServiceId,
   [out]  UInt32              DelayForNextCall,
   [out]  Int32                     ServiceError);

CancelTransfer(
   [in]   ByteString          CommunicationRelationId,
   [in]   UInt32              ServiceId,
   [out]  Int32                     ServiceError);
```

Table 39 – Transfer Method arguments

Argument	Description
CommunicationRelationId	See 8.2.4.2.
<SendData>	This is a placeholder for a protocol specific list of arguments as described in IEC 62769-4:~~2020~~2023, Annex F. The argument values represent the protocol specific communication service request that is sent to the device.
<ReceiveData>	This is a placeholder for a protocol specific list of arguments as described in IEC 62769-4:~~2020~~2023, Annex F. The argument values represent the protocol specific communication service response that is received from the device.
ServiceId	The service transaction code establishes the relation between the service request and the corresponding response.
DelayForNextCall	The value specifies a delay time in ms, the caller shall wait before the next invocation of EndTransfer.
ServiceError	0: OK / function started asynchronously, result has to be polled with EndTransfer 1: OK / execution finished, ReceivedData contains the result -1: Transfer Failed / cancelled by caller -2: Call Failed / unknown service ID -3: Transfer Failed / no existing communication relation -4: Transfer Failed / invalid communication relation identifier -5: Transfer Failed / invalid sendData content -6: Transfer Failed / invalid receiveData format

8.3 FDI® Communication Package

8.3.1 General

Subclause 8.3 specifies the FDI® Communication Package details that are specific for Gateways. The definitions given in 5.1 apply also.

8.3.2 EDD

8.3.2.1 General

The definitions in 5.2 apply. Additionally, the EDD elements as specified in IEC 62769-4 and provided with a Gateway specific FDI® Communication Package shall contain:

a) a PROFILE (IEC 61804-3): The PROFILE definition shall be chosen according to the protocol used for communication service requests;

b) a Business Logic: The communication service provider related EDD COMPONENTs shall implement the methods specified in IEC 61804-3. These methods implement the protocol bridging logic. The translation procedures open out into outbound communication requests via the EDDL Builtin library function invocation or the writing of data onto an online node. The set of usable EDDL Builtin library functions is bound to the PROFILE;

c) a Module management: The Gateway related Component can implement the ValidateModules (see 5.2.9). The implemented logic shall validate individual changes and handle any parameter related dependencies for the whole Gateway device. The implementation of ValidateModules is optional when the actual product specific COMPONENT declaration is sufficient to configure the module setup without any additional Business Logic.

8.3.2.2 Gateway Component

Each FDI® Communication Package describing a Gateway shall contain an EDD element describing the Gateway as defined in 5.2.2. Gateway specifics are described in the following:

```
COMPONENT <DeviceComponentId>
{
    LABEL "<Label>";
    CAN_DELETE TRUE;
    CHECK_CONFIGURATION <ValidateModules>;
    CLASSIFICATION NETWORK_COMPONENT;
    COMPONENT_RELATIONS
    {
       <CommunicationDeviceRelationId>
    }
    PROTOCOL <ProtocolId>;
}
```

<ProtocolID>: The existence of this attribute indicates the connectivity of the Gateway regarding outbound communication. It allows the FDI® Server to find the network using the same type of protocol to which this Gateway can be connected. For standardized protocols, the value is defined by the related fieldbus organization.

8.3.2.3 Gateway CommunicationDevice Component

Each FDI® Communication Package describing a Gateway shall contain at least one EDD element describing at least one CommunicationDevice component as defined in 5.2.3.

NOTE A Gateway is sometimes referred to as "Remote IO". Remote IO is a Modular Device supporting multiple various module types that can be flexibly assigned to any slot. Thus it is possible to create multiple different Remote IO configurations (n – slots X m – module types).

The rules about COMPONENT attribute settings shown in 5.2.3 apply. Gateway specifics are described in the following:

```
COMPONENT <CommunicationDeviceComponentId>
{
    LABEL "<Label>";
    CAN_DELETE <CanDelete>;
    CLASSIFICATION NETWORK_COMPONENT;
    COMPONENT_RELATIONS
    {
<CommunicationServiceProviderRelationId>
    }
    SCAN <Scan>;
    SCAN_LIST <ScanList>;
}
```

<Scan>: The attribute refers to the METHOD implementing the device discovery service. The reference works because the identifier value of the METHOD matches with attribute value. (Implementation details are specified in 8.2.3.2.)

<ScanList>: The attribute value refers to the LIST describing the topology scan results. This list shall contain all devices discovered during execution of the device discovery service. (Implementation details are specified in 8.2.3.4.)

8.3.2.4 Communication Service component

The rules defined in 5.2.4 apply.

Additionally, the file describing the component contains the implementations of the methods defined in 8.2.4.2, 8.2.4.3 and 8.2.4.4. The identifier used for the related METHOD constructs matches with the method names specified in Table 37.

8.3.2.5 Connection Point Component

The rules defined in 5.2.5 apply.

8.3.2.6 Connection Point Collection

The rules defined in 5.2.6 apply.

8.4 Handling and behaviour

8.4.1 General

A Gateway provides functionality to communicate between two communication protocols. Gateways are used to communicate from one network of the automation system to another (subordinated) network. Figure 16 shows a typical example where a HART®[2] device is connected to a PROFIBUS®[3] Remote I/O. In order to communicate to the HART device, the Remote I/O receives a communication request via the PROFIBUS network (see Figure 16, key 1). The communication request contains the necessary information that allows the gateway to create the according HART Command and send it to the HART device (see Figure 16, key 2).

---

2 HART® is a registered trademark of the non-profit organization FieldComm Group, Inc. This information is given for convenience of users of this document and does not constitute an endorsement by IEC of the trademark holder or any of its products. Compliance does not require use of the trademark. Use of the trademark requires permission of the trademark holder.

3 PROFIBUS® is the registered trademark of the non-profit organization PROFIBUS Nutzerorganisation e.V. (PNO). This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the trademark holder or any of its products. Compliance does not require use of the trademark. Use of the trademark requires permission of the trademark holder.

The way the HART Command is wrapped into the PROFIBUS communication request ~~may~~ can be standard or gateway specific. The HART response (see Figure 16, key 3) from the device is embedded as a PROFIBUS communication response (see Figure 16, key 4). The way the Gateway wraps the response ~~may~~ can either be standard or Gateway specific.

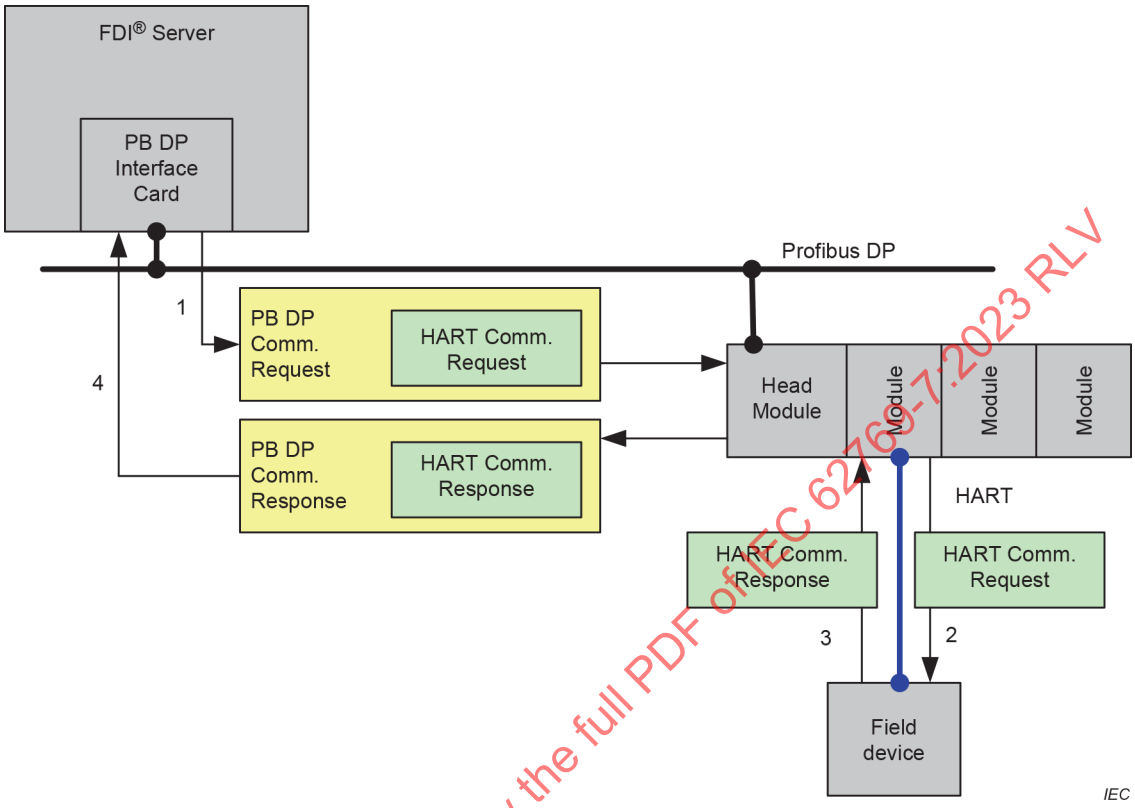


Figure 16 – Nested Communication

Subclause 8.4 defines the Gateway handling and behaviour rules along the life cycle beginning with the deployment, start up, bus commissioning, until the communication services processing.

8.4.2 Deployment

The definitions of 5.2.11 apply.

8.4.3 Start up

The Information Model and the FDI® Package EDD element based Gateway representation does not require any start up procedures.

8.4.4 Configure the communication hardware

The handling and behaviour matches with the specification in 7.8.9. The only difference is in the channel setup that is represented in the FDI® Server hosted Information Model only.

8.4.5 Configure the Network

The handling and behaviour matches with the specification in 7.8.10.

8.4.6 Parameterize

The Gateway can require proper parameter adjustment prior to any communication service processing. The FDI® Communication Package contained user dialogs (UID or UIP) enable

interactive bus parameter adjustment. The FDI® Communication Package can provide additional Business Logic for bus parameterization purposes.

8.4.7 Communication relation

The status machine definitions in Clause 6 apply.

If the Communication Gateway supports multiple instances of GatewayCommunicationServiceType, these instances need to share information about existing communication relations.

8.4.8 Connect

Prior to running any information exchange related communication, the Gateway requires establishing a communication relation between the device application and the physical network connected device. This happens through invocation of the method Connect. This enables the Gateway to perform an optional outbound communication service request, which might be needed for a specific Gateway device to establish a communication relation.

The Gateway shall be able to manage multiple communication relations. After successful execution of the method Connect, the corresponding communication relation enters the status "CR Online".

Invocation of the services Transfer and Scan is allowed in the status "Online" only.

8.4.9 Disconnect

Invocation of the method Disconnect terminates a communication relation, which inhibits further information exchange related communication with the physical network connected device.

After execution of the method Disconnect the corresponding communication relation enters the status "CR Offline". The communication relation becomes invalid.

8.4.10 Abort indication

Depending on protocol specifics, the Gateway can detect communication aborts. Such communication abort indications are returned as communication service results during the processing of the methods Transfer or Scan. After the Gateway has returned an Abort Indication, the current communication relation enters the status "CR Offline". The communication relation becomes invalid.

8.4.11 Scan

Gateways declare their device discovery service capability based on the EDDL construct COMPONENT – the attributes SCAN and SCAN_LIST. Related definitions are defined in IEC 61804-3. The SCAN attribute setting is mandatory within the EDD element describing the Gateway CommunicationDevice. The attribute value refers to the METHOD executing the topology scan function. The SCAN_LIST attribute setting is mandatory within the Gateway CommunicationDevice COMPONENT. The attribute value refers to the element that contains the topology scan result created by the method referenced by the attribute value SCAN. The SCAN_LIST shall refer to a LIST containing COLLECTION elements describing the detected devices (Scan-List-Item). The protocol specific content of a Scan-List-Item is described in IEC 62769-4:~~2020~~2023, Annex F.

The invocation of the functions Scan and ScanNext results in outbound communication service requests.

8.4.12 Communication Error Handling

The communication service processing shall use the EDDL Builtin function provided abort according to the EDDL Profiles:

- during the creation of communication messages,
- during the processing of the response from the communication request.

The communication service processing does not trigger communication errors based on communication timeouts.

Annex A
(informative)

Layered protocols

A.1 General

Ethernet based protocols commonly consist of a stack of different protocols based on the ISO/OSI model. Looking at the growing number of Ethernet based fieldbus protocols, it is crucial to have a common layered modeling concept for Connection Points based on the ISO/OSI model also.

Connection Points are the elements that contain the address information accessed by the Communication Devices to collect the information needed for communication. The semantics of Connection Point attributes need to be standardized.

The PROFINET®[4] device ~~may~~ can concurrently support PROFINET, SNMP and HTTP. The information stored in the Connection Point of a device is different for each protocol due to different application layers. The information for layer 1 to 4 shall consistently hold the same information. Therefore, Connection Points shall inherit Connection Point information from lower network layers.

The problem is about how to ensure that address information from lower layers is named in a consistent way throughout all protocols that are built upon lower layers.

A.2 Convention for protocol specific annex creation

A.2.1 General

Since Connection Point description is based on EDDL, an actual inheritance approach known from object-oriented programming languages does not seem applicable. The approach described in Clause A.2 is based more on conventions. The naming convention shall ensure that the address value attribute names defined for a "lower level" protocol are reused in the Connection Point definitions for higher level protocols. The same holds true for the COLLECTION referred VARIABLES. VARIABLE declarations for higher-level protocols shall be copies from the VARIABLE declarations for lower-level protocols. The convention described here is applicable for the creation of protocol specific annexes.

A.2.2 Connection Point

The following shows some EDDL examples of how Connection Point declarations follow this naming convention.

```
COLLECTION ConnectionPoint_MAC
{
    LABEL "<Label>";
    MEMBERS
    {
        MAC,
        VALID
    }
```

[4] PROFINET® is the registered trademark of the non-profit organization PROFIBUS Nutzerorganisation e.V. (PNO). This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the trademark holder or any of its products. Compliance does not require use of the trademark. Use of the trademark requires permission of the trademark holder.

```
}



COLLECTION ConnectionPoint_IPv4
{
    LABEL "<Label>";
    MEMBERS
    {
        MAC,
        IPv4,
        VALID
    }
}



COLLECTION ConnectionPoint_TCPUDP
{
    LABEL "<Label>";
    MEMBERS
    {
        MAC,
        IP,
        PORT,
        VALID
    }
}



COLLECTION ConnectionPoint_PROFINET
{
    LABEL "<Label>";
    MEMBERS
    {
        MAC,
        IP,
        PORT,
        DNSNAME,
        VALID
    }
}



COLLECTION ConnectionPoint_HTTP
{
    LABEL "<Label>";
    MEMBERS
    {
        MAC,
        IP,
        PORT,
        URL,
        VALID
    }
}
```

A.3 FDI® Communication Package definition

A.3.1 Communication services

The actual communication service is always implemented according to the specific protocol. This is reflected by the different semantic of the communication service arguments specified by the protocol specific annexes. So the actual implementation of a ServerCommunicationDeviceType or GatewayCommunicationDeviceType can support just one protocol. Thus, if an FDI® Communication Server or Gateway is able to support multiple different protocols, it shall describe separate GatewayCommunicationDeviceTypes or ServerCommunicationDeviceTypes. The need to define protocol specific service sets represents the demand to separate the Connection Point and Network related COMPONENT definitions described in A.3.2 and A.3.3.

A.3.2 Connection Point

The FDI® Communication Package shall contain separate Connection Point descriptions for each supported protocol.

A.3.3 Network

The relation between the COMPONENT describing the network and the COMPONENT describing the Connection Point enables generic communication path detection and generic topology configuration. Thus the FDI® Communication Package shall contain a separate COMPONENT definition for each supported Network (protocol).

A.4 Representation in the Information Model

Connection Points sharing a certain set of address formation ~~may~~ can contain redundant address information, for example the IP address is the same for the SNMP and PROFINET I/O.

If a Device and an FDI® Communication Server share a set of protocols, then that Device and FDI® Communication Server are associated through multiple separate networks.

A Device supporting multiple protocols can be connected to different FDI® Communication Devices that support only one protocol.

Annex B
(normative)

Namespace and Mappings

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

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

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

The BrowsePath for an Instance Node is constructed by appending the BrowseName of the Instance Node to the BrowseName for the containing instance or type. An underscore character is used to separate each BrowseName in the path.

The NamespaceUri http://FDI-cooperation.com/OpcUa/FDI7/ is applied to NodeIds defined here.

The CSV released with this version of the document is provided by FieldComm Group™, ~~see https://www.fieldcommgroup.org~~. An electronic version of the complete Information Model defined in this document is also provided. It follows the XML Information Model Schema syntax defined in IEC 62541-6.

The Information Model Schema released with this version of the document is provided by FieldComm, Inc Group™.

IEC 62769-7

Edition 3.0 2023-04

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Field Device Integration (FDI®) – Part 7: Communication Devices

Intégration des appareils de terrain (FDI®) – Partie 7: Appareils de Communication

IEC 62769-7:2023-04(en-fr)

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

FIELD DEVICE INTEGRATION (FDI®) –

Part 7: Communication Devices

FOREWORD

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

IEC 62769-7 has been prepared by subcommittee 65E: Devices and integration in enterprise systems, of IEC technical committee 65: Industrial-process measurement, control and automation. It is an International Standard.

This third edition cancels and replaces the second edition published in 2021. This edition constitutes a technical revision.

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

a) added ScanExtended Method.

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

Draft	Report on voting
65E/859/CDV	65E/916/RVC

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

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

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

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

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

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

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

FIELD DEVICE INTEGRATION (FDI®) –

Part 7: Communication Devices

1 Scope

This part of IEC 62769 specifies the elements implementing communication capabilities called Communication Devices.

The overall FDI®[1] architecture is illustrated in Figure 1. The architectural components that are within the scope of this document have been highlighted in this illustration. The document scope with respect to FDI® Packages is limited to Communication Devices. The Communication Server shown in Figure 1 is an example of a specific Communication Device.

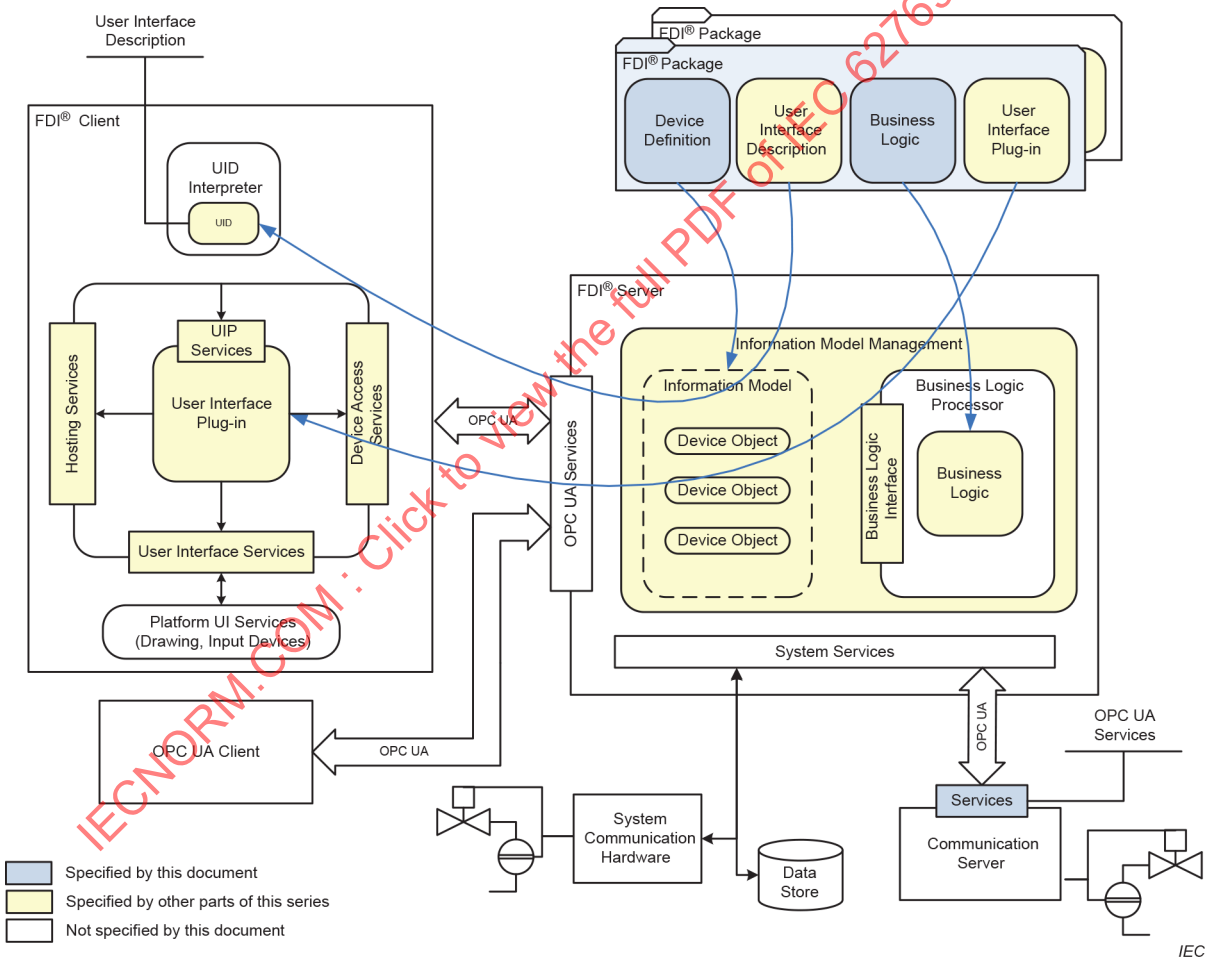


Figure 1 – FDI® architecture diagram

[1] FDI® is a registered trademark of the non-profit organization Fieldbus Foundation, Inc. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the trademark holder or any of its products. Compliance does not require use of the trade name. Use of the trade name requires permission of the trade name holder.

2 Normative references

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

IEC 61804-3, *Devices and integration in enterprise systems − Function blocks (FB) for process control and electronic device description language (EDDL) − Part 3: EDDL syntax and semantics*

IEC 61804-4, *Devices and integration in enterprise systems − Function blocks (FB) for process control and electronic device description language (EDDL) − Part 4: EDD interpretation*

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

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

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

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

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

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

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

IEC 62769-3, *Field Device Integration (FDI®) – Part 3: Server*

IEC 62769-4:2023, *Field Device Integration (FDI®) – Part 4: FDI® Packages*

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

3 Terms, definitions, abbreviated terms, acronyms and conventions

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 62769-1, IEC 62769-3, IEC 62541-6 and the following apply.

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

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

3.1.1
Gateway
Communication Device that enables to bridge between different physical networks or different protocols

3.2 Abbreviated terms and acronyms

For the purposes of this document, the abbreviated terms and acronyms given in IEC 62769-1, IEC 62541-6 and the following apply.

HTTP Hypertext Transfer Protocol

IP Internet Protocol

PHY Physical communication hardware

SNMP Simple Network Management Protocol

TCP Transmission Control Protocol

URI Uniform Resource Identifier

3.3 Conventions

3.3.1 EDDL syntax

This part of IEC 62769 specifies content for the EDD component that is part of FDI® Communication Packages. The specification content using EDDL syntax uses the font `Courier New`. The EDDL syntax is used for method signature, variable, data structure and component declarations.

3.3.2 Capitalizations

Capitalization of the first letter of words is used in the IEC 62769 series to emphasize an FDI® defined term.

3.3.3 Graphical notation

This document uses the graphical notation defined in IEC 62769-5.

4 Overview

The abstract term FDI® Communication Device represents an entity implementing communication functions over a network using a specific protocol. The group of FDI® Communication Devices splits into two main groups.

a) The FDI® Communication Server is a dedicated OPC UA Server providing access to one or more field device networks. The FDI® Communication Server is specified in Clause 7.

b) The FDI® Communication Gateway enables to bridge between different physical networks or different protocols. The bridging business logic is implemented in the EDD component that is provided with an FDI® Communication Package. The FDI® Communication Gateway is specified in Clause 8.

NOTE The main differences between a Gateway and a Communication Server are as follows:
In terms of FDI® the FDI® Communication Server is a dedicated OPC UA Server providing access to one or more field device networks. A Gateway is a Communication Device that enables to bridge between different physical networks or different protocols. The logical representation of a Gateway device within the FDI® Server hosted Information Model enables the FDI® Server to process communication in heterogeneous network topologies.

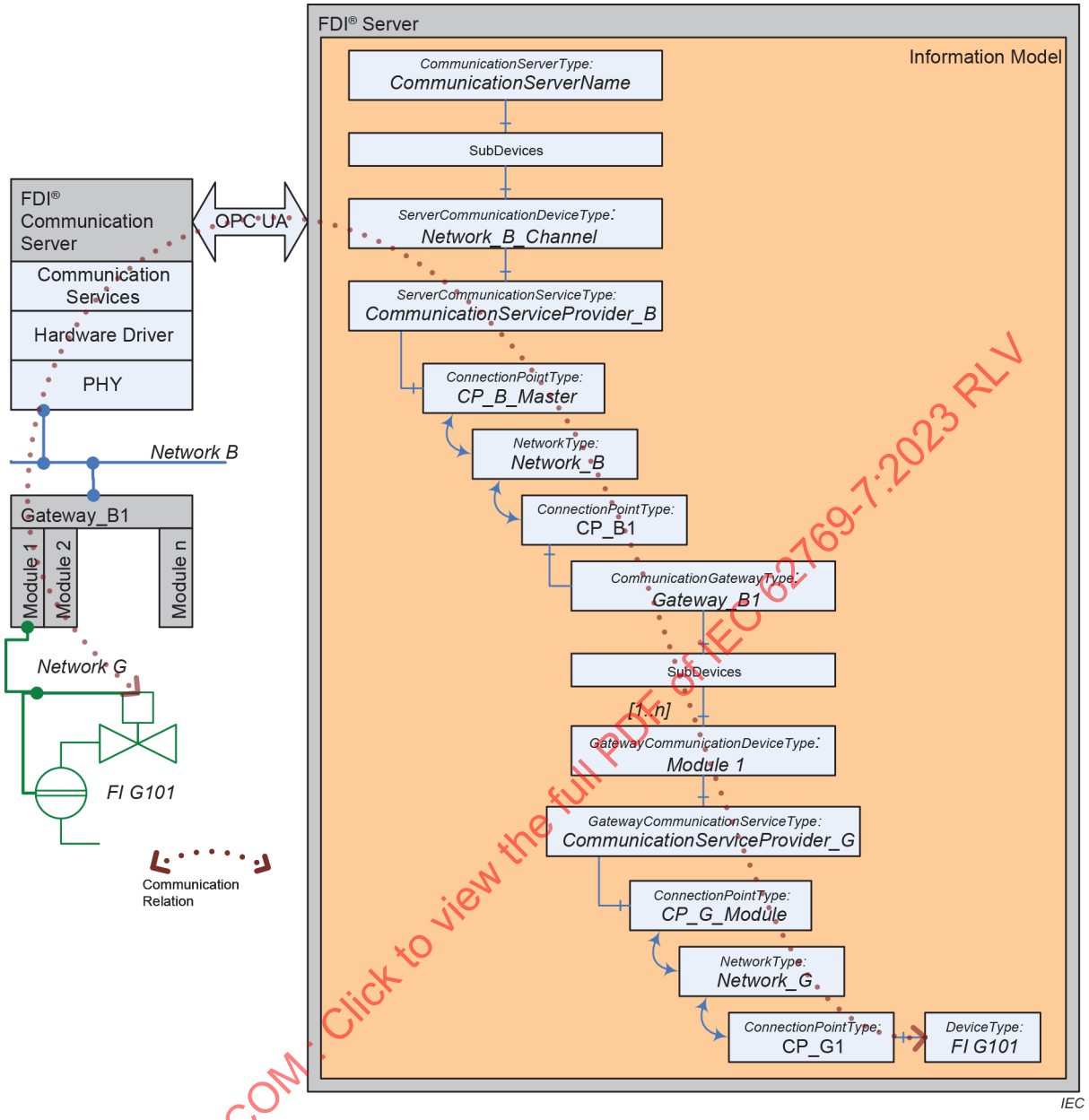


Figure 2 – FDI® communication infrastructure architecture

The FDI® Server hosted Information Model contains a representation of the network topology. (see also IEC 62769-5). The Information Model shown in Figure 2 is an example excerpt to illustrate how the Information Model used elements reflect the actual network topology.

1) The instance of CommunicationServerType (named CommunicationServerName) represents the FDI® Communication Server. The FDI® Communication Server implements physical communication network access (Communication hardware). Clause 7 describes related Information Model specifics, required FDI® Communication Package content and handling of elements therein. (For subdevices, see IEC 62769-5)
2) The instance of ServerCommunicationDeviceType and ServerCommunication-ServiceType (named Network_B_Channel) maps to the FDI® Communication Server implemented communication services. The ServerCommunicationDeviceType is specified in 7.3.3. The ServerCommunicationServiceType is specified in 7.3.4.
3) The instance of CommunicationGatewayType (named Gateway_B1) represents the physical Gateway. Clause 8 describes the related Information Model specifics, the required FDI® Package content and the handling of elements therein.

4) The instance of GatewayCommunicationDeviceType (named Module 1) maps to a physical or logical module enabling communication to the network to which this module is connected. The GatewayCommunicationDeviceType is specified in 8.3.2.3. The related Gateway specifics are described in Clause 8.

5) The instance of GatewayCommunicationServiceType (named CommunicationServiceProvider_G) represents the Gateways' ability to process communication services. The Gateway specific implementation of GatewayCommunicationServiceType is based on Business Logic that enables to run communication services in heterogeneous communication networks.

6) A communication relation (more details are described in Clause 6) between a physical device and the device representation managed by the FDI® Server is always associated to communication service objects that are instances of a GatewayCommunicationServiceType or ServerCommunicationServiceType. The ability of instantiating multiple communication service objects supports protocols enables to operate multiple logical connections between a bus master and a device.

7) The Information Model represents the connections between the physical devices shown on the left side of Figure 2 based on instances of ConnectionPointType NetworkType and the depicted relations. ConnectionPointType and NetworkType are specified in IEC 62541-100.

5 FDI® Communication Package

5.1 General

The FDI® Server imports the FDI® Communication Package like any other FDI® Device Package. Clause 5 specifies the FDI® Communication Package details.

5.2 EDD

5.2.1 General rules

The FDI® Communication Package contained EDD is not restricted but bound to a protocol specific profile (see IEC 62769-4:2023, Annex F).

The EDD elements as specified in the protocol specific profile documents (see IEC 62769-4:2023, Annex F), and provided with an FDI® Communication Package shall describe:

a) Parameter and parameter structures. Mandatory protocol specific parameter definitions are found in the protocol specific profile documents (see IEC 62769-4: 2023, Annex F). The parameters shall contain any parameter that requires adjustment for proper communication service operation.

b) Physical Layer identification. Protocol specific definitions are found in IEC 62769-4: 2023, Annex F.

c) Communication Devices modularity: The modularity information shall be based on using the EDDL constructs COMPONENT (see IEC 61804-3).

   FDI® envisions Communication Device modularity to cope with communication hardware providing multiple physical or logical communication channels to access multiple logical or physical communication networks. Each module element of the whole Communication Device shall be described by a separate EDD element.

d) The COMPONENT definition shall be used to support the system implemented topology configuration. Protocol specific definitions are found in the protocol specific profile documents (see IEC 62769-4:2023, Annex F). The related COMPONENT definitions are described in 5.2.2, 5.2.3, 5.2.4, and 5.2.7.

e) The Business Logic shall contain a method enabled to validate the network (see 5.2.8). The validation function considers the elements only directly connected to the network. The validation function shall be referred by the EDDL specified CHECK_CONFIGURATION attribute.

f) The Business Logic can contain a method enabled to validate the module configuration (see 5.2.9) or the network configuration (see 5.2.8). The validation function considers the elements only directly connected to the related parent element in the topology. The validation function shall be referred by the EDDL specified CHECK_CONFIGURATION attribute.

g) Connection Point data: The Connection Point (see 5.2.4 and 5.2.6) shall be described through EDDL constructs COMPONENT, COLLECTION and VARIABLE. The COMPONENT definition associates the Connection Point element to the Communication Device. The VARIABLE definitions represent the properties of a specific Connection Point. The COLLECTION represents the Connection Point structure as such. Protocol specific definitions are found in IEC 62769-4:2023, Annex F. Annex A describes a convention for protocol specific annex creation.

h) MENU:

The Menu structure shall follow the Menu conventions for PC based applications according to IEC 61804-4, enabling access to

1) FDI® Communication Device Type (Bus) parameters: These parameters shall be made accessible by means of ""offline_root_menu";

2) Topology Configuration Dialogs shall be made available by means of the menu entry point "topology_configuration".

5.2.2 Device component

Each FDI® Communication Package shall contain an EDD element describing the device.

```
COMPONENT <DeviceComponentId>
{
     LABEL "<Label>";
     CAN_DELETE TRUE;
     CHECK_CONFIGURATION <ValidateModules>;
     CLASSIFICATION NETWORK_COMPONENT;
     COMPONENT_RELATIONS
     {
        <CommunicationDeviceRelationId>
     }
}



COMPONENT_RELATION <CommunicationDeviceRelationId>
{
    LABEL "Relation type description";
    RELATION_TYPE CHILD_COMPONENT;
    ADDRESSING {<AddressVar>}
    COMPONENTS
    {
       <CommunicationDeviceComponentId>
        {
          AUTO_CREATE <autoCreate>;
          REQUIRED_RANGES
            {
              <AddressVar>{ MIN_VALUE <AddrMin>; MAX_VALUE <AddrMax>;}
    }
}
    }
    MINIMUM_NUMBER <minNumber>;
    MAXIMUM_NUMBER <maxNumber>;
}
```

<DeviceComponentId>: The COMPONENT identifier identifies the component description for the device type.

<Label>: The string value shall contain a string that allows a human user to determine the function of the FDI® Communication Server object.

<ValidateModules>: The Value refers to the METHOD implementing the module topology configuration validation function. (Implementation details are specified in 5.2.9.)

The attribute COMPONENT_RELATIONS allows to describe how modules can be connected. The definition of the COMPONENT_RELATIONS is optional. If used, it shall describe the relations to the CommunicationDevice definitions. The construct enables to perform generic, FDI® Server driven (device) topology configuration. Syntax details are described in IEC 61804-3. The subsequent text describes the semantic use of the COMPONENT_RELATION construct.

<CommunicationDeviceRelationId>: The attribute value identifies the COMPONENT_RELATION definition describing the relation between the device component and the CommunicationDevice component.

<CommunicationDeviceComponentId>: The attribute value needs to match with a COMPONENT identifier used in a COMPONENT declaration that describes a CommunicationDevice (see 5.2.3).

<autoCreate>: The attribute value describes the number of CommunicationDevice components that can be automatically instantiated with the Device component.

<minNumber>/<maxNumber>/<autoCreate>: The attribute values define the instantiation constraints. The definition of these attributes is optional. The attribute values can contain conditional expressions.

The RELATION_TYPE shall be set to CHILD_COMPONENT.

<AddressVar>: The attribute value is a reference to a VARIABLE declaration. This VARIABLE holds the address value for a CommunicationDevice instance. The definition of this attribute is optional.

<AddrMin>/<AddrMax>: Values define the address value range for a CommunicationDevice instance. The value can, for example, correspond to a physical slot number. Usage of attributes ADDRESSING and REQUIRED_RANGES enables generic configuration routines.

5.2.3 CommunicationDevice component

Each FDI® Communication Package shall contain at least one EDD element describing at least one CommunicationDevice component. A modular communication hardware structure shall be described by multiple CommunicationDevice COMPONENT descriptions:

```
COMPONENT <CommunicationDeviceComponentId>
{
    LABEL "<Label>";
    CAN_DELETE <CanDelete>;
    CLASSIFICATION NETWORK_COMPONENT;
    COMPONENT_RELATIONS
    {
<CommunicationServiceProviderRelationId>
    }
}
```

```
COMPONENT_RELATION <CommunicationServiceProviderRelationId>
{
   LABEL "Relation between CommunicationDevice and communication
service provider";
   RELATION_TYPE CHILD_COMPONENT;
   ADDRESSING {<AddressVar>}
   COMPONENTS
   {
      <CommunicationServiceProviderId>
      {
         AUTO_CREATE <autoCreate>;
}
   }
   MINIMUM_NUMBER 1;
   MAXIMUM_NUMBER <maxNumber>;
}
```

<CommunicationDeviceComponentId>: The COMPONENT identifier identifies the CommunicationDevice component.

<Label>: The string value shall contain a human readable string that allows a user to easily determine the function of the CommunicationDevice component.

<CanDelete>: Allowed values are TRUE or FALSE. It depends on whether a CommunicationDevice needs explicit configuration or whether the related communication service provider object shall be automatically instantiated with the CommunicationDevice. If the attribute CAN_DELETE is set to FALSE, the CommunicationDevice configuration is static.

The definition of the COMPONENT_RELATIONS is mandatory. It describes the relation to the communication service provider definition. The construct enables the FDI® Server to instantiate communication service provider components according to communication processing demands. (Syntax details are described in IEC 61804-3.) The subsequent text describes the semantic use of the COMPONENT_RELATION construct.

<CommunicationServiceProviderRelationId> The attribute value identifies the COMPONENT_RELATION definition as such.

<CommunicationServiceProviderId>: The attribute value has to match with a COMPONENT identifier used in a COMPONENT declaration that describes a communication service provider (5.2.4).

<autoCreate>: The attribute value describes the number of communication service providers that can be automatically instantiated with the CommunicationDevice component.

The RELATION_TYPE shall be set to CHILD_COMPONENT.

The PROTOCOL attribute shall not be set.

5.2.4 Communication service provider component

Each FDI® Communication Package describing a Communication Device shall contain at least one EDD element describing the communication service provider. The EDD component shall not define any configuration parameter.

```
COMPONENT <CommunicationServiceProviderId>
{
    LABEL "<Label>";
    BYTE_ORDER <byteOrder>;
    CAN_DELETE <CanDelete>;
    CLASSIFICATION NETWORK_COMMUNICATION_SERVICE_PROVIDER;
    COMPONENT_RELATIONS
    {
<CommunicationServiceProvidersConnectionPointRelationId>
    }
}

COMPONENT_RELATION
<CommunicationServiceProvidersConnectionPointRelationId>
{
   LABEL "Relation between communication service provider and
connection point";
   RELATION_TYPE CHILD_COMPONENT;
   ADDRESSING {<AddressVar>}
   COMPONENTS
   {
      < ConnectionPointId>
      {
         AUTO_CREATE 1;
}
   }
   MINIMUM_NUMBER 1;
   MAXIMUM_NUMBER 1;
}
```

<CommunicationServiceProviderId>: The COMPONENT identifier identifies the communication service provider.

<Label>: The string value shall contain a human readable string that allows a user to easily determine the function of the communication service provider object.

<CanDelete>: Allowed values are TRUE or FALSE. It depends on whether a communication service provider can be flexibly instantiated according to the communication processing demands. If the attribute CAN_DELETE is set to FALSE, the number of communication service provider component instantiations is static. The instantiation constraints declared through the attributes AUTO_CREATE, MINIMUM_NUMBER and MAXIMUM_NUMBER correspond to the capabilities of currently supported protocols.

<byteOrder>: The value enables generic integration of n-byte data types (e.g. 4-byte Integer) into the communication message payload. The attribute value describes the byte order and shall be either BIG_ENDIAN or LITTLE_ENDIAN.

The definition of the COMPONENT_RELATIONS is mandatory. It describes the relation to the Connection Point definition. The construct enables to perform generic, FDI® Server driven topology configuration. (Syntax details are described in IEC 61804-3.) The subsequent text describes the semantic use of the COMPONENT_RELATION construct.

The Connection Point shall automatically be instantiated with the communication service provider and there shall be exactly one (1) Connection Point instance connected to the communication service provider. The instantiation constraints declared through the attributes AUTO_CREATE, MINIMUM_NUMBER and MAXIMUM_NUMBER correspond to the capabilities of currently supported protocols.

<CommunicationServiceProvidersConnectionPointRelationId> The attribute value identifies the COMPONENT_RELATION declaration as such.

<ConnectionPointId>: The attribute value has to match with a COMPONENT identifier used for a COMPONENT declaration that describes a Connection Point (see 5.2.5).

The RELATION_TYPE shall be set to CHILD_COMPONENT.

The PROTOCOL attribute shall not be set.

5.2.5 Connection Point component

Each FDI® Communication Package describing a Communication Device shall contain one EDD element describing one Connection Point for each of the protocols that are supported by the Communication Device:

```
COMPONENT <ConnectionPointId>
{
    LABEL "<Label>";
    CAN_DELETE FALSE;
    CLASSIFICATION NETWORK_CONNECTION_POINT;
    PROTOCOL <ProtocolId>;
    CONNECTION_POINT <ConnectionPointCollectionId>;
}
```

<ConnectionPointId>: The COMPONENT identifier identifies the Connection Point component declaration.

<Label>: The string value shall contain a string that allows a human user to determine the function of the Connection Point component.

<ProtocolID>: The value of this attribute indicates the communication capability which allows the FDI® Server to find other device types that can be connected to the network using the same type of protocol. For standardized protocols, the value is defined by the related field bus organization.

<ConnectionPointCollectionId>: The attribute value is a reference to a COLLECTION declaration that describes the data structure of the Connection Point as described in 5.2.6 .

5.2.6 Connection Point collection

Each EDD describing the Connection Point of a Communication Device shall describe the COLLECTION element that describes the attributes that shall appear in the Information Model representation of the Connection Point. The protocol specific data exposed by the Connection Point identifies the device type and its network address.

```
COLLECTION <ConnectionPointCollectionId>
{
    LABEL "<Label>";
    MEMBERS
    {
        <AddressAttributeName>, <AddressAttributeVariableId>;
        VALID <VALID_VariableId>;
    }
}
```

<ConnectionPointCollectionId>: The identifier of the COLLECTION is referred by the CONNECTION_POINT attribute value defined in 7.7.3.5.

<Label>: The label identifies the Connection Point in a human readable way.

<AddressAttributeName>/<AddressAttributeVariableId>: The MEMBER section refers to the VARIABLE definitions describing the address attributes implemented by a Connection Point. The content of the MEMBER section is protocol specific.

<VALID>/<VALID_VariableId> is a Collection member referring a Boolean VARIABLE holding the validation status that shall be set by the ValidateNetwork Action (see 5.2.8).

5.2.7 Network component

Each FDI® Communication Package describing a Communication Device shall contain one EDD element describing one Network for each of the protocols that are supported by the Communication Device. The definition supports the network topology engineering:

```
COMPONENT <NetworkComponentId>
{
    LABEL "<Label>";
    CAN_DELETE TRUE;
    CHECK_CONFIGURATION <Validate>;
    CLASSIFICATION NETWORK;
    PROTOCOL <ProtocolId>;
    COMPONENT_RELATIONS
    {
        <NetworksConnectionPointRelationId>
    }
}



COMPONENT_RELATION <NetworksConnectionPointRelationId>
{
   LABEL "Relation between network and connection point";
   RELATION_TYPE CHILD_COMPONENT;
   ADDRESSING {<AddressVar>}
   COMPONENTS
   {
      <ConnectionPointId>
      {
         REQUIRED_RANGES
           {
             <BusAddressVar>{ MIN_VALUE <BusAddrMin>; MAX_VALUE
<BusAddrMax>;}
   }
}
   }
   MINIMUM_NUMBER 1;
   MAXIMUM_NUMBER <maxNumber>;
}
```

<NetworkComponentId>: The COMPONENT identifier identifies the Network component declaration.

<Label>: The string value shall contain a human readable string that allows a user to easily determine the function of the Network component.

<Validate>: The Value refers to the METHOD implementing the network topology configuration validation function (see 5.2.8).

<ProtocolID>: The value of this attribute allows the FDI® Server to find other device types that can be connected to the network using the same type of protocol. For standardized protocols, the value is defined by the related fieldbus organization.

The definition of the COMPONENT_RELATIONS is mandatory. It describes the relation to the Connection Point definition and by that the capabilities of a network. The construct enables to perform generic, FDI® Server driven network topology configuration. Syntax details are described in IEC 61804-3. The subsequent text describes the semantic use of the COMPONENT_RELATION construct.

<NetworksConnectionPointRelationId> The attribute value identifies the COMPONENT_RELATION definition.

<ConnectionPointId>: The attribute value has to match with a COMPONENT identifier used for a COMPONENT declaration that describes a Connection Point (see 5.2.4).

<maxNumber: The attribute value limits the number of Connection Points that can be connected to the network. The attribute values can contain conditional expressions.

The RELATION_TYPE shall be set to CHILD_COMPONENT.

<BusAddressVar>: The attribute value is a reference to a VARIABLE declaration. This VARIABLE holds the network address value for any device that is connected to the network.

<BusAddrMin>/<BusAddrMax>: Values define the network address value range.

5.2.8 ValidateNetwork

The method ValidateNetwork represents the Communication Device implemented Business Logic that validates a current network topology. The ValidateNetwork method handles any necessary dependencies related to bus parameters. The implementation of related EDDL logic is based on the EDDL Builtin function ObjectReference, which enables to analyze a set of child instances (Connection Point instances). The validation logic shall set the <VALID> attribute of the Connection Point instance that has passed the validation.

The implementation of ValidateModules is optional if the module setup is either static or if the configuration rules defined in the COMPONENT construct are sufficient to configure the module setup.

Table 1 shows the ValidateNetwork Action arguments.

Signature

```
ValidateNetwork(
  [out] Int32  ServiceError,
  [out] String ErrorMessage);
```

Table 1 – ValidateNetwork Action arguments

Argument	Description
ServiceError	0: OK -1: Failed / the Connection Point that did not pass the validation is indicated by the <VALID> attribute () value set to false. Remark: The argument values correspond to the IEC 61804-3 specified error codes named BI_SUCCESS (value = 0) and BI_ERROR (value = -1). The Action returns the ServiceError result using the "return" statement.
ErrorMessage	If the method returns an empty string (NULL) the Action call succeeded. In case of an error, the Action can return a problem description.

5.2.9 ValidateModules

The method ValidateModules validates the current module setup. The implementation of the related EDDL logic is based on the EDDL Builtin function ObjectReference, which enables to analyze a set of child instances. The implementation of ValidateModules is optional if the module setup is either static or if the configuration rules defined in the COMPONENT construct are sufficient to configure the module setup.

NOTE The decision whether ValidateModules is needed or not is vendor specific.

Table 2 shows the ValidateModules Action Arguments.

Signature

```
ValidateModules(
  [out] Int32  serviceError,
  [out] String ErrorMessage);
```

Table 2 – ValidateModules Action arguments

Argument	Description
ServiceError	0: OK -1: Failed / the Connection Point that did not pass the validation is indicated by the <VALID> attribute () value set to false. Remark: The argument values correspond to the IEC 61804-3 specified error codes named BI_SUCCESS (value = 0) and BI_ERROR (value = -1). The Action returns the ServiceError result using the "return" statement.
ErrorMessage	If the Action returns an empty string (NULL), the method call succeeded. In case of an error, the Action can return a problem description.

5.2.10 UIP specifics

The FDI® Communication Package can contain the UIP to support e.g. diagnostics and parameterization.

5.2.11 Deployment

The FDI® Server imports the FDI® Communication Package. The handling of EDD and UIP parts matches with the import procedure performed for the FDI® Package (see IEC 62769-2 and IEC 62769-3).

6 Communication relations

The purpose of a Communication Device and its communication services is to exchange information between the physical device and the device representation managed by the FDI® Server. The information exchange is managed via communication relations, see Figure 3. An established communication relation represents the capability to exchange information between the FDI® Server managed device representation and the physical device. The use of a Communication relation allows abstracting from protocol specifics typically used to manage connections.

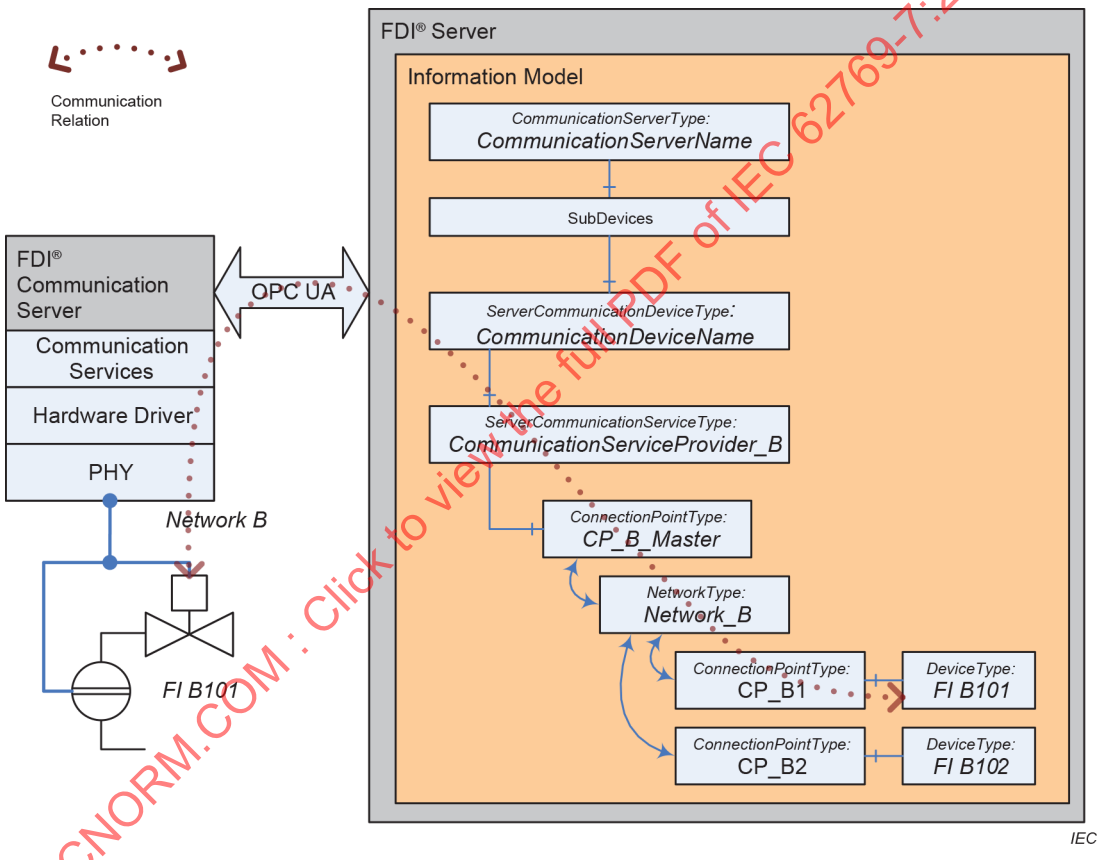


Figure 3 – Communication relation

NOTE 1 The core of information exchange happens between the physical network connected device and the corresponding instance within the Information Model but does not cover the complete device application.

The following state chart describes the general state flow for a single communication relation. The diagram also shows which communication services can be invoked during a "CR Online" state.

The "AbortIndication" shown in Figure 4 can be detected in different protocol specific ways. The one specified for any Communication Device is bound to the serviceErrors returned by the specified communication services. Even the Scan Method can determine a connection loss, when the device for which a communication relation has been activated does not appear in a scan result.

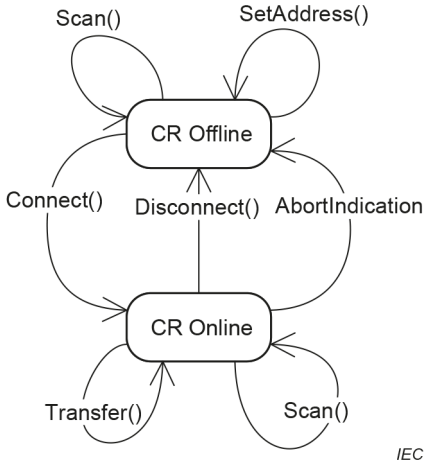


Figure 4 – Communication relation state chart

NOTE 2 The management of communication relations is optional.

7 FDI® Communication Server definition

7.1 General

In terms of FDI®, the FDI® Communication Server is a dedicated OPC UA Server providing access to one or more field device networks. Each FDI® Communication Server is modelled as a Modular Device where each module (also called CommunicationDevice in the sequence) represents the access point to one network.

The Modular Device itself represents the FDI® Communication Server as a whole.

7.2 General characteristics

The FDI® Communication Server implements characteristics for each of its CommunicationDevices specified in 7.3.3. Additionally, an FDI® Communication Server implements the following characteristics:

- The FDI® Server always synchronizes (see 7.5, 7.8.8, and 7.8.11) the FDI® Communication Server hosted Information Model from the FDI® Server hosted Information Model content.
- CommunicationDevices can be statically instantiated or they can be created/deleted by the FDI® Server.
- Communication between the FDI® Server and the FDI® Communication Server is based on OPC UA. OPC UA specifies a wire protocol for its services that can be implemented on arbitrary platforms and runtime environments.
- To avoid race conditions, the FDI® Communication Server only allows one FDI® Server being connected at a time. With this restriction, an FDI® Communication Server can refrain from any synchronization (locking) mechanism. The FDI® specification does not enforce FDI® Communication Servers implementing any interlocking mechanism to manage concurrent access to a single physical network connected device.

7.3 Information Model

7.3.1 General

Subclause 7.3 specifies the FDI® Communication Server hosted Information Model.

An FDI® Communication Server is an OPC UA Server that encapsulates communication hardware and provides standardized communication ability. The FDI® Server connects to the FDI® Communication Server as an OPC UA Client and accesses the networks supported by the

FDI® Communication Server via the FDI® Communication Server Information Model. The task of the FDI® Communication Server is to expose this Information Model. The FDI® Communication Server shall not maintain Device Instances or network topology information. All interaction with FDI® devices is done through the FDI® Server and just transferred by the FDI® Communication Server.

For the FDI® Server, an FDI® Communication Server looks like a device that supports FDI® Communication Services and uses OPC UA to communicate. The FDI® Communication Server can run locally on the same PC as the FDI® Server (loop back adapter) or remote in the field (e.g., embedded into a controller). Like a device, each FDI® Communication Server has an associated FDI® Package. This FDI® Package is used to create Communication Devices in the Information Model of the FDI® Server that represent access to the networks implemented by the FDI® Communication Server.

The Information Model of an FDI® Communication Server is based on the Information Model defined in IEC 62769-5. Figure 5 replicates the Modular Device structure and illustrates how it maps into the overall AddressSpace. The modules represent the communication channels of the FDI® Communication Server. Figure 5 defines the BrowseNames for the nodes.

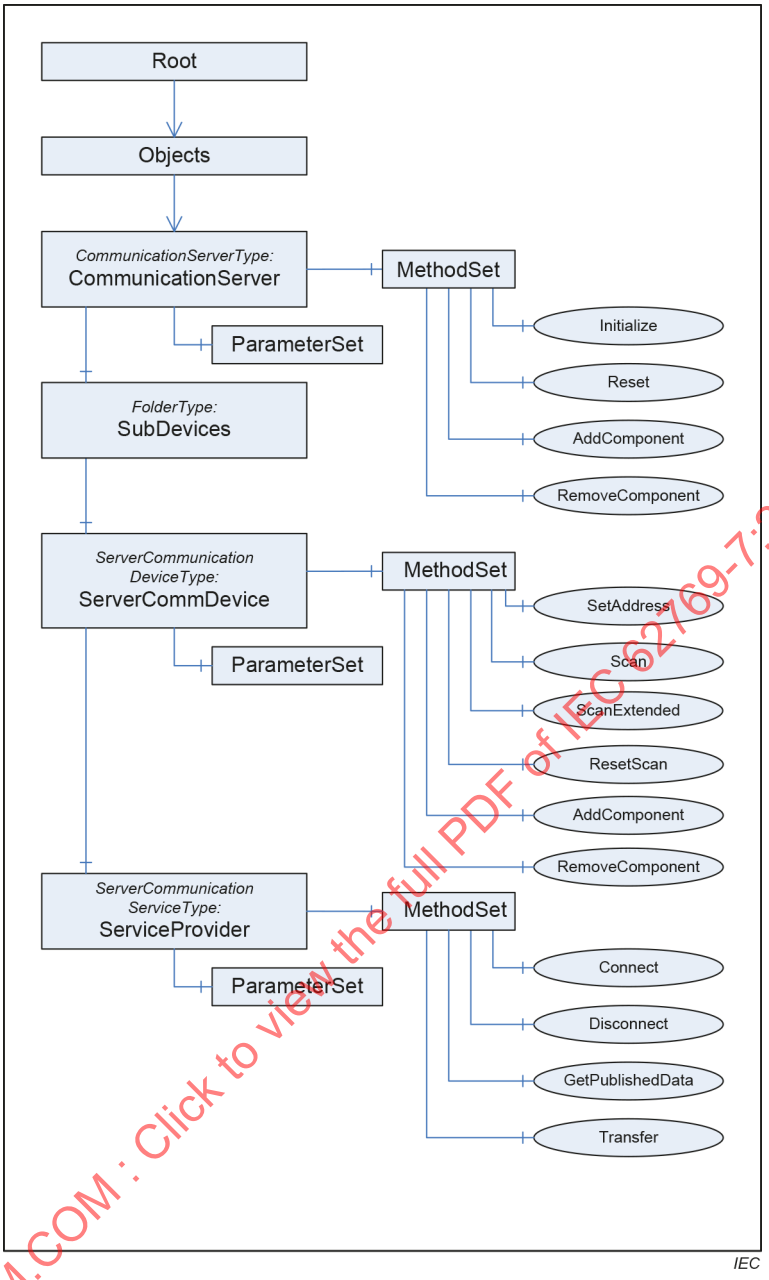


Figure 5 – FDI® Communication Server AddressSpace

The CommunicationServerType (the root of the Modular Device) is a subtype of the DeviceType. The MethodSet contains the methods Initialize, Reset, AddComponent and RemoveComponent. The methods AddComponent and RemoveComponent are optionally present if the FDI® Communication Server supports the dynamic instantiation of elements in the folder SubDevices.

All sub devices are instances of the ServerCommunicationDeviceType defined in 7.3.3. The instances of the ServerCommunicationDeviceType (ServerCommDevice) have a MethodSet that can implement the methods SetAddress, Scan, AddComponent, RemoveComponent. AddComponent and RemoveComponent are optionally present if the FDI® Communication Server supports a variable number of instances of the ServerCommunicationServiceType.

Formal definitions are found in 7.3.2, 7.3.3 and 7.3.4.

7.3.2 CommunicationServerType

7.3.2.1 General

The CommunicationServerType is a subtype of the DeviceType and provides the methods needed to manage the instances ServerCommunicationDeviceType. Figure 6 shows the CommunicationServerType definition that is formally defined in Table 3 and Table 4.

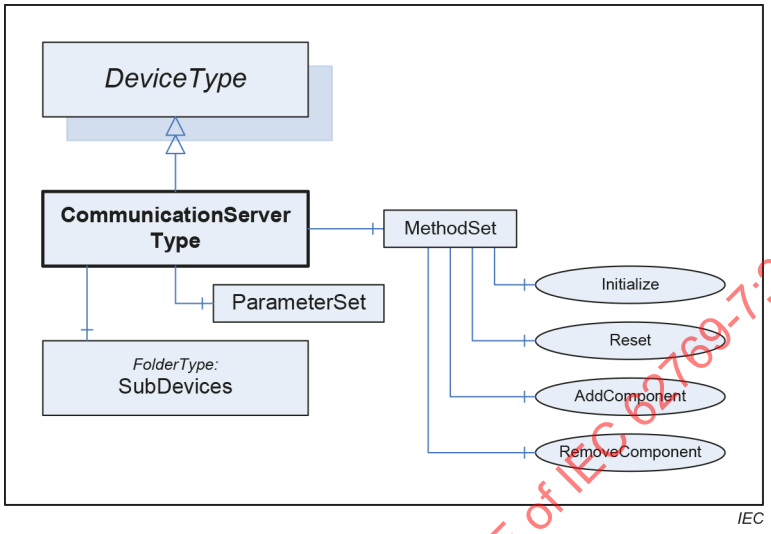


Figure 6 – CommunicationServerType

Table 3 – CommunicationServerType definition

Attribute	Value				
BrowseName	CommunicationServerType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the DeviceType defined in IEC 62541-100.					
HasComponent	Object	MethodSet		BaseObjectType	Mandatory
HasComponent	Object	ParameterSet		BaseObjectType	Optional
HasComponent	Object	SubDevices		FolderType	Mandatory

Table 4 – MethodSet of CommunicationServerType

Attribute	Value				
BrowseName	MethodSet				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
HasComponent	Method	Initialize			Mandatory
HasComponent	Method	Reset			Mandatory
HasComponent	Method	AddComponent			Optional
HasComponent	Method	RemoveComponent			Optional

The CommunicationServerType and each instance of this Type share the same Methods. The NodeId of these Methods will be fixed and defined in this document (see Annex B). FDI®

Communication Server clients therefore do not have to browse for these Methods. They can use the fixed NodeId as the MethodId of the Call Service.

The additional Methods AddComponent and RemoveComponent add the ability to add or remove instances of ServerCommunicationDeviceType according to communication hardware structure. These services are not applicable if the FDI® Communication Server implements a static communication hardware structure.

The SubDevices folder contains instances of ServerCommunicationDeviceType that represent the communication modules.

NOTE The indication for a static communication hardware layout is indicated in the FDI® Package with COMPONENT attribute CAN_DELETE set to FALSE in COMPONENT declarations.

7.3.2.2 Reset Method

Reset is used to reset the communication hardware and related driver software. Any ongoing communication will be stopped immediately. All communication channels enter the status closed.

The Method Reset shall not be present in the FDI® Server hosted Information Model. The FDI® Server shall be able to handle the shut-down procedure automatically according to communication demands.

Typically, the FDI® Communication Server operation includes some hardware and protocol driver handling that can be independent from any modular structure. Because of this possibility, the Reset Method is arranged underneath the CommunicationServerType. For the purpose of reducing the complexity of FDI® Communication Server operation, only one Reset Method has been specified.

The signature of this Method is specified below. Table 5 and Table 6 specify the arguments and AddressSpace representation, respectively.

Signature

```
Reset(
  [out] Int32  serviceError);
```

Table 5 – Reset Method arguments

Argument	Description
serviceError	0: OK -1: Failed

Table 6 – Reset Method AddressSpace definition

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

7.3.2.3 Initialize Method

Initialize is used to initialize the communication hardware. The initialization function of the FDI® Communication Server shall use the parameterization data hosted by the ParameterSet that is

contained within the instance of the CommunicationServerType and all instances of ServerCommunicationDeviceType.

In order to enable parameter changes during operation, the Initialize method can be re-invoked. If the FDI® Communication Server needs to reset its communication hardware, it shall automatically restore any communication relation that existed. A modular FDI® Communication Server can flexibly initialize only those ServerCommunicationDeviceType instances for which configuration changes have been detected.

The Method Initialize shall not be present in the FDI® Server hosted Information Model. The FDI® Server shall be able to handle the start procedure automatically according to human driven communication requests.

The FDI® Communication Server operation can include some hardware and protocol driver handling that can be independent from any modular structure. Because of this possibility, the Initialize method is arranged underneath the CommunicationServerType. For the purpose of reducing the complexity of FDI® Communication Server operation, only one Initialize method has been specified.

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

Signature

```
Initialize(
  [out] Int32  serviceError)
```

Table 7 – Initialize Method arguments

Argument	Description
serviceError	0: OK -1: Failed

Table 8 – Initialize Method AddressSpace definition

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

7.3.2.4 AddComponent Method

AddComponent shall be used to configure the modular setup of an FDI® Communication Server in case the FDI® Communication Sever has no statically defined communication hardware setup. This method shall be used to add a module (Instance of ServerCommunicationDeviceType).

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

Signature

```
AddComponent(
  [in]   String   ModuleTypeName,
  [in]   String   InstanceName,
  [in]   String   InstanceLabel,
  [out]  NodeId   InstanceNodeId,
  [out]  Int32    ServiceError);
```

Table 9 – AddComponent Method arguments

Argument	Description
ModuleTypeName	Type of module to be created as defined in the FDI® Package. The module type name shall correspond to one of the COMPONENT identifier definitions (see 5.2.3).
InstanceName	Non-localized name of the module's Device Node of the created element. This name has to be unique within the scope of the FDI® Communication Server's Information Model.
InstanceLabel	Human readable label for the root Node of the created module.
InstanceNodeId	Callee-assigned identifier for the module's Device Node.
ServiceError	0 – OK -1 – E_InvalidType – a module for the specified Type cannot (not anymore) be added -2 – E_DuplicateName – there exists already a module with the same name as specified with the InstanceName argument -3 – E_UnknownType – an unknown ModuleTypeName has been specified -4 – E_LimitExceeded – the total number of modules is exceeded (this might be caused by power constraints or other resource limitations)

Table 10 – AddComponent Method AddressSpace definition

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

7.3.2.5 RemoveComponent Method

RemoveComponent shall be used to remove a module (Instance of ServerCommunicationDeviceType). Implementation of RemoveComponent is optional if the communication hardware setup is static.

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

Signature

```
RemoveComponent(
  [in]  NodeId     ModuleNodeId,
  [out] Int32      ServiceError);
```

Table 11 – RemoveComponent Method arguments

Argument	Description
ModuleNodeId	The value is the identification of the existing instance in the Information Model.
ServiceError	0: OK -1: Failed, the specified node does not exist

Table 12 – RemoveComponent Method AddressSpace definition

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

7.3.3 ServerCommunicationDeviceType

7.3.3.1 General

The ServerCommunicationDeviceType represents a communication channel for a particular network. The ServerCommunicationDeviceType is a subtype of the DeviceType. The ParameterSet for each instance of a ServerCommunicationDevice will contain Parameters necessary to configure the operation of the network. Figure 7 shows the ServerCommunicationDeviceType definition that is formally defined in Table 13 and Table 14.

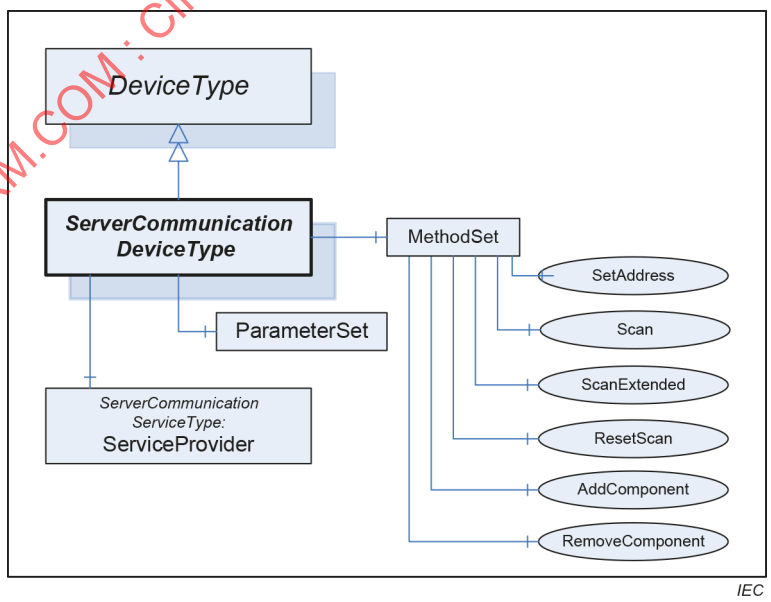


Figure 7 – ServerCommunicationDeviceType

Table 13 – ServerCommunicationDeviceType definition

Attribute	Value				
BrowseName	ServerCommunicationDeviceType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the DeviceType defined in OPC UA Part DI.					
HasComponent	Object	MethodSet		BaseObjectType	Optional
HasComponent	Object	ParameterSet		BaseObjectType	Optional
HasComponent	Object	ServiceProvider		ServerCommunicationServiceType	Mandatory
HasProperty	Variable	ListOfCommunicationProfiles	String[]	PropertyType	Mandatory

The Property ListOfCommunicationProfiles contains a list of communication profiles supported by the ServerCommunicationDevice. Valid strings are defined in IEC 62769-4:2023, Annex F.

Table 14 – MethodSet of ServerCommunicationDeviceType

Attribute	Value				
BrowseName	MethodSet				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
HasComponent	Method	Scan			Optional
HasComponent	Method	ScanExtended			Optional
HasComponent	Method	ResetScan			Optional
HasComponent	Method	SetAddress			Optional
HasComponent	Method	AddComponent			Optional
HasComponent	Method	RemoveComponent			Optional

7.3.3.2 Scan Method

Scan shall be used to start discovering physical network connected devices. The associations between the method Scan and the corresponding physical network connection enables the FDI® Communication Server to access the correct physical network connection. The Scan Method is implemented by the Communication Server runtime module.

The signature of this Method is specified below. Table 15 and Table 16 specify the arguments and AddressSpace representation, respectively.

NOTE 1 Communication Servers can run the network scan in a background task so the invocation of the function Scan will return cached network scan results.

NOTE 2 In case the SCAN takes very long, the FDI® Communication Server might return an empty TopologyScanResult and the ServiceError 1 identifying that the scan is still running.

Signature

```
Scan(
  [out] XmlElement      TopologyScanResult,
  [out] Int32           ServiceError)
```

Table 15 – Scan Method arguments

Argument	Description
TopologyScanResult	The argument value is an XML formatted string representing a list of physical network connected devices. Each of the physical network connected devices is represented by a data structure matching with a Connection Point node. Connection Point attributes are protocol specific. The corresponding topologyScanResult schema is specified in IEC 62769-4:2023, Annex F. Return an empty string for TopologyScanResult in case of any error.
ServiceError	0: OK / scan completed 1: OK / get complete scan result by calling Scan again -1: Failed / not initialized -2: Failed / not connected to a network -3: Failed / no device found, the topologyScanResult is empty

Table 16 – Scan Method AddressSpace definition

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

7.3.3.3 ScanExtended Method

ScanExtended shall be used to start discovering physical network connected devices, if the FDI® Server and the FDI® Communication Server support at least FDI® Technology Version 1.3. This method performs the same functionality as the Scan Method. It returns more detailed information about the devices found on the network. The ScanExtended Method is implemented by the Communication Server runtime module. ScanExtended is only available if the communication profile requires the method.

The signature of this Method is specified below. Table 17 and Table 18 specify the arguments and AddressSpace representation, respectively.

NOTE 1 Communication Servers can run the network scan in a background task so the invocation of the function Scan will return cached network scan results.

NOTE 2 In case the SCAN takes very long, the FDI® Communication Server might return an empty TopologyScanResultExtended and the ServiceError 1 identifying that the scan is still running.

Signature

```
Scan(
  [out] XmlElement      TopologyScanResultExtended,
  [out] Int32           ServiceError)
```

Table 17 – Scan Method arguments

Argument	Description
TopologyScanResultExtended	The argument value is an XML formatted string representing a list of physical network connected devices. Each of the physical network connected devices is represented by a data structure matching with a Connection Point node. Connection Point attributes are protocol specific. The corresponding topologyScanResultExtended schema is specified in IEC 62769-4:2023, Annex E. Return an empty string for TopologyScanResultExtended in case of any error.
ServiceError	0: OK / scan completed 1: OK / get complete scan result by calling Scan again -1: Failed / not initialized -2: Failed / not connected to a network -3: Failed / no device found, the topologyScanResult is empty

Table 18 – Scan Method AddressSpace definition

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

7.3.3.4 ResetScan Method

ResetScan shall be used to reset the internal cache of scan results. It will also cancel a running scan in case the FDI® Communication Server scan mechanism supports this.

The signature of this Method is specified below. Table 19 and Table 20 specify the arguments and AddressSpace representation, respectively.

Signature

```
ResetScan(
  [out] Int32      ServiceError)
```

Table 19 – ResetScan Method arguments

Argument	Description
ServiceError	0: OK / scan reset -1: Failed / not initialized -2: Failed / not connected to a network

Table 20 – ResetScan Method AddressSpace definition

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

7.3.3.5 SetAddress Method

SetAddress shall be used to change the network address (communication address) of a device. The Communication Device shall ensure unique network address values. If the argument value of newAddress is already assigned to a physical network connected device the Communication Device shall return the argument serviceError value "-4: Failed / duplicate Address error".

It depends on the protocol whether the address assignment service shall work even when a communication relation is already established.

The signature of this Method is specified below. The arguments for SetAdress Method are described in Table 21.

Signature

```
SetAddress(
    [in]                        <OldAddress>,
    [in]                        <NewAddress>,
    [out]  Int32                ServiceError);
```

Table 21 – SetAddress Method arguments

Argument	Description
<OldAddress>	This is a placeholder for a 1..n protocol specific arguments representing the existing protocol specific network address of the physical network connected device. Values that represent a network address are specified in protocol specific profile documents (see IEC 62769-4:2023 Annex F).
<NewAddress>	This is a placeholder for 1..n protocol specific arguments representing the new protocol specific network address that shall be assigned to a physical network connected device. Values that represent a network address arguments are specified in IEC 62769-4:2023, Annex F.
ServiceError	0: OK / execution finished successfully Other values that represent protocol specific ServiceErrors might be specified in protocol specific profile documents (see IEC 62769-4:2023, Annex F).

7.3.4 ServerCommunicationServiceType

7.3.4.1 General

Communication services provide the means to communicate with a Device or to e.g. execute a Scan on some Network. Communication services are represented through Methods in the Information Model (see IEC 62769-5).

The formal definition of ServerCommunicationServiceType is found in Figure 8, Table 22 and Table 23.

The NodeId of these Methods will be fixed and defined in this document (see Annex B). FDI® Clients therefore do not have to browse for these Methods. They can use the fixed NodeId as the MethodId of the Call Service.

Communication methods including their NodeIds are uniquely defined in this document. FDI® Clients can use the Methods directly (without browsing). The OPC UA Call Service shall be used as follows:

- the MethodId argument shall contain the fixed NodeId of the Method;
- the ObjectId argument shall contain the NodeId of the MethodSet.

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

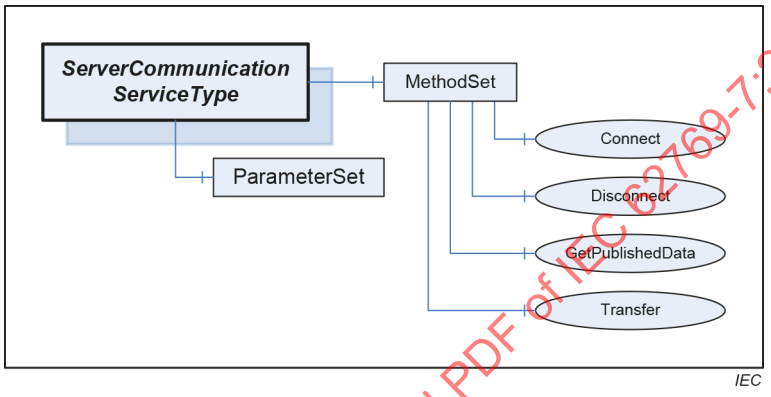


Figure 8 – ServerCommunicationServiceType

Table 22 – ServerCommunicationServiceType definition

Attribute	Value				
BrowseName	ServerCommunicationServiceType				
IsAbstract					
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the DeviceType defined in OPC UA Part DI.					
HasComponent	Object	MethodSet		BaseObjectType	Mandatory
HasComponent	Object	ParameterSet		BaseObjectType	Optional

Table 23 – MethodSet of ServerCommunicationServiceType

Attribute	Value				
BrowseName	MethodSet				
IsAbstract					
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
HasComponent	Method	Connect			Optional
HasComponent	Method	Disconnect			Optional
HasComponent	Method	Transfer			Mandatory
HasComponent	Method	GetPublishedData			Optional

7.3.4.2 Connect Method

Connect shall be used to establish a communication relation to a device that is physically connected to the network. Establishing the communication relation can imply checks of identification data that are part of the addressData with data inside the physical device. The Communication Device performs this DeviceType match verification according to a corresponding network protocol standard. Related details are specified in IEC 62769-4:2023, Annex F.

The devices address is contained in the Connection Point of the corresponding Device Instance within the Information Model (Device Connection Point). The communication relation between the Information Model associated device application and the physical device is further on identified by the communication relation identifier. Details about how to manage the status of a communication relation is described in Clause 6.

NOTE 1 As the NodeId is a unique identifier within the Information Model scope, the NodeId of the Device Connection Point can be a unique identifier for any communication relation in the scope of a Communication Device.

NOTE 2 The term communication relation is introduced describing the status of an infrastructure that enables data exchange between information model hosted data and a physical device. If the communication relation is established data exchange is possible.

The signature of this Method is specified below. Table 24 specifies the arguments.

Signature

```
Connect(
   [in]   ByteString        CommunicationRelationId,
   [in]                     <AddressData>,
   [out]                    <DeviceInformation>,
   [out]  Int32             ServiceError);
```

Table 24 – Connect Method arguments

Argument	Description
CommunicationRelationId	This is a client generated id that is used to uniquely identify this connection. This could be an index (e.g., a NodeId) that the client (= FDI® Server) needs to identify entries in its topology.
<AddressData>	This is a placeholder for a protocol specific argument list that is used for the address and optional device identification data (details described in IEC 62769-4:2023, Annex F.
<DeviceInformation>	This is a placeholder for a protocol specific argument list in which the connect result data are stored.
ServiceError	0: OK / execution finished, connection established successfully Other values that represent protocol specific ServiceErrors might be specified in protocol specific profile documents (see IEC 62769-4:2023, Annex F)

7.3.4.3 Disconnect method

Disconnect shall be used to terminate a communication relation to a Device.

The signature of this Method is specified below. Attributes of the Disconnect method are specified in Table 25. Disconnect is a synchronous method call.

Signature

```
Disconnect(
   [in]   ByteString        CommunicationRelationId,
   [out]  Int32             ServiceError);
```

Table 25 – Disconnect Method arguments

Argument	Description
CommunicationRelationId	Same ID as used in method Connect specified in 7.3.4.2.
ServiceError	1: OK / disconnect finished successfully Other values that represent protocol specific ServiceErrors might be specified in protocol specific profile documents (see IEC 62769-4:2023, Annex F)

7.3.4.4 Transfer method

Transfer shall be used to perform information exchange with a Device.

The signature of this Method is specified below. All arguments are specified in Table 26.

Signature

```
Transfer(
   [in]   ByteString            CommunicationRelationId,
   [in]                              <SendData>,
   [out]                             <ReceiveData>,
   [out]  Int32                      ServiceError);
```

Table 26 – Transfer Method arguments

Argument	Description
CommunicationRelationId	See 7.3.4.2.
<SendData>	This is a placeholder for a protocol specific list of arguments as described in IEC 62769-4:2023, Annex F. The argument values represent the protocol specific communication service request that is sent to the device.
<ReceiveData>	This is a placeholder for a protocol specific list of arguments as described in IEC 62769-4:2023, Annex F. The argument values represent the protocol specific communication service response that is received from the device.
ServiceError	0: OK / execution finished, ReceivedData contains the result Other values that represent ServiceErrors are specified in IEC 62769-4:2023, Annex F.

7.3.4.5 GetPublishedData Method

The FDI® Server sends GetPublishedData requests to the FDI® Communication Server to receive data that is submitted by unsolicited data messages. The argument SendData contained data prepares the exchange of "unsolicited" data messages from the device. The content of SendData is protocol specific. The FDI® Communication Server queues GetPublishedData requests in a queue associated with the Communication Relation defined through the argument CommunicationRelationId. The argument PublishId identifies the related queue entry. Each time the FDI® Communication Server receives unsolicited data messages, it saves the received data in association with the existing queue entry that has been created for the GetPublishedData. Depending on the underlying network technology (performance), the method GetPublishedData can immediately return with data coming from an "unsolicited" data message.

Subsequent pulling of data that is submitted by unsolicited data messages works through the same method GetPublishedData. In this case, the argument SendData is empty. The argument PublishId matches with the value that has been provided with the initial call GetPublishedData that has established the transmission of exchange of "unsolicited" data messages.

In order to stop the device sending the "unsolicited" data, the method GetPublishedData shall be used again but the argument SendData contained data terminates the exchange of "unsolicited" data messages from the device. Table 27 shows the GetPublishedData Method arguments.

Signature

```
GetPublishedData(
   [in]   ByteString            CommunicationRelationId,
   [in]                         <SendData>,
   [out]                        <ReceiveData>,
   [out]  DateTime              TimeStamp
   [in]   UInt32                PublishId,
   [out]  Int32                 ServiceError);
```

Table 27 – GetPublishedData Method arguments

Argument	Description
CommunicationRelationId	See 7.3.4.2.
<SendData>	This is a placeholder for a protocol specific list of arguments as described in IEC 62769-4:2023, Annex F. The argument values control the exchange of unsolicited messages.
<ReceiveData>	This is a placeholder for a protocol specific list of arguments as described in IEC 62769-4:2023, Annex F. The argument values convey data that comes from unsolicited messages.
TimeStamp	Time, when the data was published by the device
PublishId	The number identifies an established subscription that conveys data that comes from unsolicited messages.
ServiceError	0: OK / execution finished Other values that represent protocol specific ServiceErrors might be specified in protocol specific profile documents (see IEC 62769-4:2023, Annex F)

7.4 OPC UA Server Profile for FDI® Communication Server

Profiles are named groupings of ConformanceUnits as defined in IEC 62541-7. The term Facet in the title of a Profile indicates that this Profile is expected to be part of another larger Profile or concerns a specific aspect of OPC UA. Profiles with the term Facet in their title are expected to be combined with other Profiles to define the complete functionality of an OPC UA Server or Client. The minimum required OPC UA Server Profile is the "Micro Embedded Device Server Profile".

The following table specifies the facet for an OPC UA Server that acts as an FDI® Communication Server. Table 28 describes Conformance Units included in this facet.

Table 28 – FDI®CommunicationServer_Facet definition

Conformance Unit	Description	Optional/ Mandatory
FDI® Comunication Server Information Model	Support at least one instance of CommunicationServerType.	M

7.5 Mapping the FDI® Server Information Model to the FDI® Communication Server IM

7.5.1 General

The representation of an FDI® Communication Server in the AddressSpace of an FDI® Server is almost identical to the AddressSpace that exists in the FDI® Communication Server. This refers in particular to the Modular Device hierarchy and the Parameters of all Devices. However, the Nodes in the FDI® Server are built from the device description imported via the FDI® Communication Package.

7.5.2 Information Model differences

Because of their different tasks, however, there are a few differences and a set of synchronization rules. The Information Model example shown in Figure 9 depicts the commonalities and the differences between the Information Models hosted by the FDI® Communication Server and the FDI® Server. In general, the FDI® Communication Server hosted Information Model is a subset of the FDI® Server hosted Information Model. The Device Instances in the FDI® Server and the FDI® Communication Server adhere to the same type definitions. Thus, browse names of common Information Model elements shall have the same browse name.

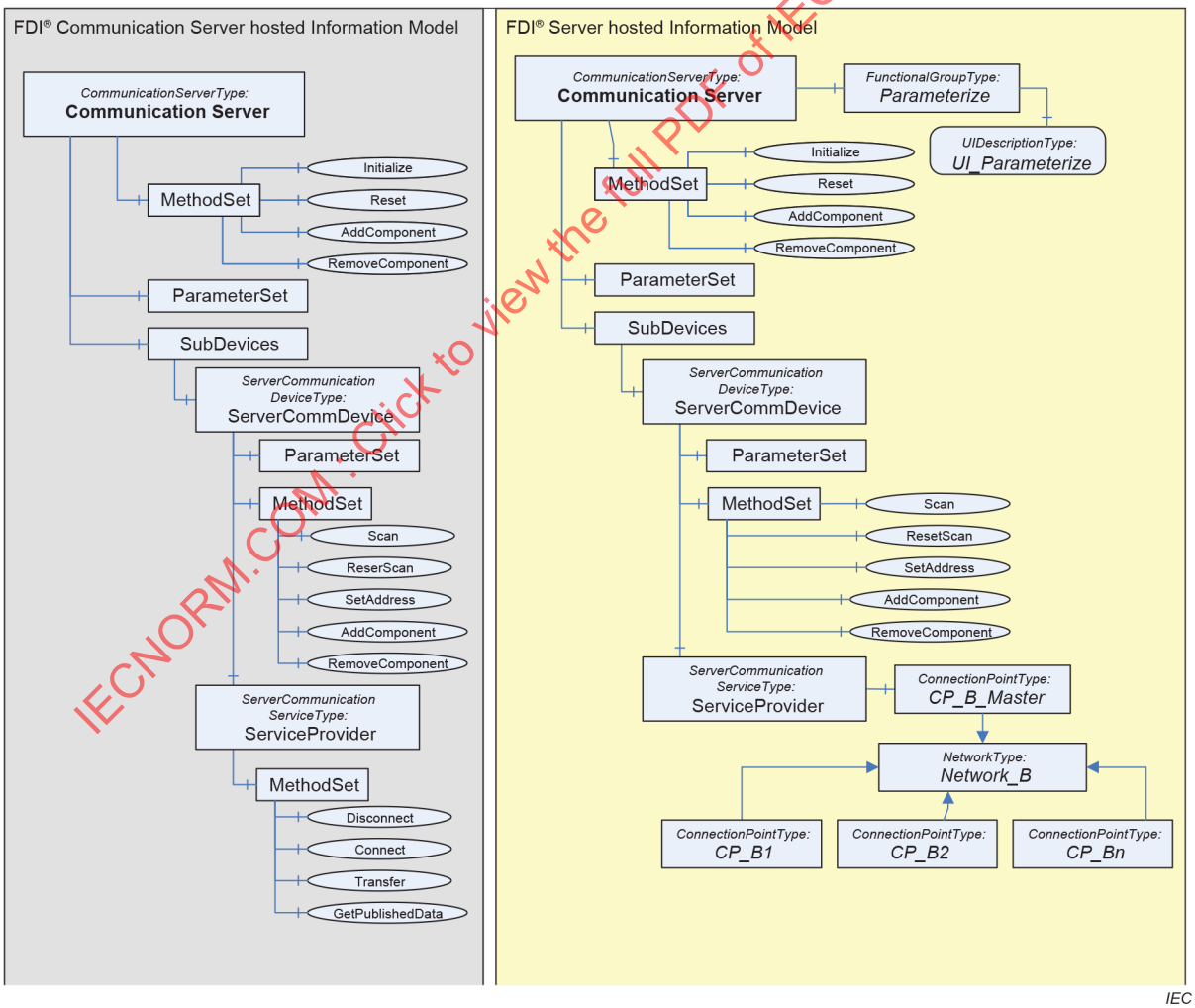


Figure 9 – Information Model differences (example)

The list of differences in the Information Model is as follows:

- The FDI® Server supports online and offline versions of the Modular Device; the FDI® Communication Server supports just an online version. The online version of the FDI® Server represents the version in the FDI® Communication Server, i.e., if Parameter Values are read or written to the online model of the FDI® Server, these operations are passed through to the FDI® Communication Server. This happens both for public and for private Parameters.

  NOTE The key is a match between the browse names present in the FDI® Server hosted Information Model and the FDI® Communication Server hosted Information Model. This allows generic synchronization of both information models.

- UIPs, UIDs, Actions and Functional Groups exist only in the FDI® Server.
- Modules in the FDI® Server hosted Information Model have a Connection Point component that is used to connect this module to a network when creating the Device Topology. The FDI® Communication Server hosted Information Model does not show Connection Point elements. The Device Topology is managed by the FDI® Server only (see IEC 62769-5).
- The FDI® Server can represent the ServiceProvider without exposing the MethodSet in order to prevent an FDI® Client from invoking communication services.

The mapping of the Module Management functionality is as follows:

- AddComponent and RemoveComponent are exposed in the FDI® Communication Server Information Model via the FDI® Communication Server Object (the root of the Modular Device). They exist only if there are modules to be configured, i.e. they will not be available if the Communication Server does not support modular communication hardware configuration. AddComponent and RemoveComponent replace the generic Node Management service defined by OPC UA.

The FDI® Server handles module topology related configuration based on the Node Management Service Set (see IEC 62769-3 and IEC 61804-4). On any module configuration related activity, the FDI® Server first calls the ValidateModules Action. The EDD Action can run through various states and even perform user dialogs (see description about Actions in IEC 62769-2 and IEC 62769-5). The EDD Methods can maintain (private) information that is global to the Modular Device. The EDD Action can access the module that shall be created.

7.6 Installer

The Installer for the FDI® Communication Server executable is optional. Since the FDI® specification does not prescribe the implementation platform for FDI® Communication Server executables, the FDI® Communication Server executable can be also preinstalled on dedicated hardware.

The installer used for the FDI® Communication Server executable shall be separated from the FDI® Package. Importing the FDI® package is a separate procedure (see 7.7.1).

7.7 FDI® Communication Package

7.7.1 General

The FDI® Server imports the FDI® Communication Package like any other FDI® Package. Subclause 7.7 specifies the FDI® Communication Package details.

With respect to the EDD element of the Package, FDI® differentiates between a simple (lightweight) Communication Server (see 7.7.2) and a regular (multi-channel) Communication Server (see 7.7.3).

7.7.2 EDD for lightweight Communication Server

A lightweight Communication Server provides access to a single field device network. It shall provide all configuration capabilities in its main EDD, not in the sub-modules used to expose the connection points. This allows FDI® Hosts not supporting modular devices to parameterize an FDI® Communication Server using the standard FDI® user interface mechanisms.

The EDD describing a "lightweight" Communication Server shall follow the IEC 61804-3 specified profile for the Communication Server. But it shall not use the following EDDL syntax constructs:

- COMPONENT,
- COMPONENT_RELATION,
- COMPONENT_FOLDER,
- COMPONENT_REFERENCE,
- EDDL Builtin function ObjectReference.

7.7.3 EDD for multi-channel Communication Server

7.7.3.1 General

The required content for an FDI® Communication Package EDD element describing an FDI® Communication Device is specified in Clause 5. Specific EDD element content for an FDI® Communication Server is described in 7.7.3.

The rules defined in 5.2.2 apply.

The PROTOCOL attribute shall not be set.

The COMPONENT declaration shall have an additional attribute PRODUCT_URI. The attribute value holds a string describing the FDI® Communication Server product URI that enables the FDI® Server to identify the FDI® Communication Server based on the OPC UA Discovery service (see IEC 61804-3). The attribute value corresponds to the RegisterServer argument RegisteredServer:serverUri. The product URI shall contain the company name and the product name.

EXAMPLE: PRODUCT_URI "urn:Company:ProductName".

7.7.3.2 CommunicationDevice component

The rules defined in 5.2.3 apply.

7.7.3.3 Communication Service component

The rules defined in 5.2.4 apply.

7.7.3.4 Connection Point component

The rules defined in 5.2.5 apply.

7.7.3.5 Connection Point collection

The rules defined in 5.2.6 apply.

7.7.4 COMMANDs in EDDs for FDI® Communication Servers

An EDD for an FDI® Communication Server shall follow the communication profile CS defined in IEC 61804-3. As the synchronization between the VARIABLEs of the EDD represented as

parameters in the FDI® Servers information model and the actual configuration of the FDI® Communication Server (represented as parameters in the information model of the FDI® Communication Server) is done by the FDI® Server, there is no need to define COMMANDs in the EDD in order to access the data of the FDI® Communication Server.

However, the communication profile CS allows the definition of COMMANDs in an EDD. This allows an EDD developer to explicitly trigger communication to the FDI® Communication Server. This can be needed in complex EDD methods that require to directly write to the FDI® Communication Server or directly read the actual value before proceeding with the method execution.

The COMMAND does not require addressing information (like NUMBER, SLOT, INDEX (see IEC 61804-3) since the addressing is done via the BrowseName of the FDI® Communication Server and the EDD identifier of the VARIABLE. Instead of the RESPONSE of a READ COMMAND shall contain the VARIABLEs to be read and the REQUEST of a WRITE COMMAND shall contain all the VARIABLEs to be written.

If such a COMMAND is called (by ReadCommand or WriteCommand Builtins in the EDD), the FDI® Server shall read respectively write all the VARIABLEs defined in the COMMAND.

Since COMMANDs also group access to VARIABLEs, a FDI® Server shall always handle VARIABLEs defined in a COMMAND as a unit. For example, when a WRITE COMMAND contains the two VARIABLEs VarA and VarB and the FDI® Server needs to write VarA because it was changed by the user, the FDI® Server also needs to write VarB in the same OPC UA service call, although VarB might not have been changed. Same applies for reading VARIABLEs.

OPC UA READ and WRITE service calls return different status codes (see IEC 62541-4). First, per call one service result code and second per parameter to be read or written one operation level result code. Since an EDD COMMAND only supports one RESPONSE_CODE per call, the following mapping applies: The first status code that is not GOOD shall be used as RESPONSE_CODE with the following order: First investigate the service result code, then the operation level result codes in the oder the VARIABLEs appear in the COMMAND definition. If all status codes are GOOD, the RESPONSE_CODE is GOOD.

7.7.5 Documentation

The FDI® Communication Package shall provide documentation describing:

a) the software installation related environment requirements and procedures,

b) the OPC UA Server configuration if needed.

7.8 Handling and behaviour

7.8.1 General

Subclause 7.8 defines the FDI® Communication Server handling and behaviour rules along the life cycle beginning with the deployment, start up, bus commissioning, until the communication services processing. The diagram (see Figure 10) shows the FDI® Server maintained FDI® Communication Server status.

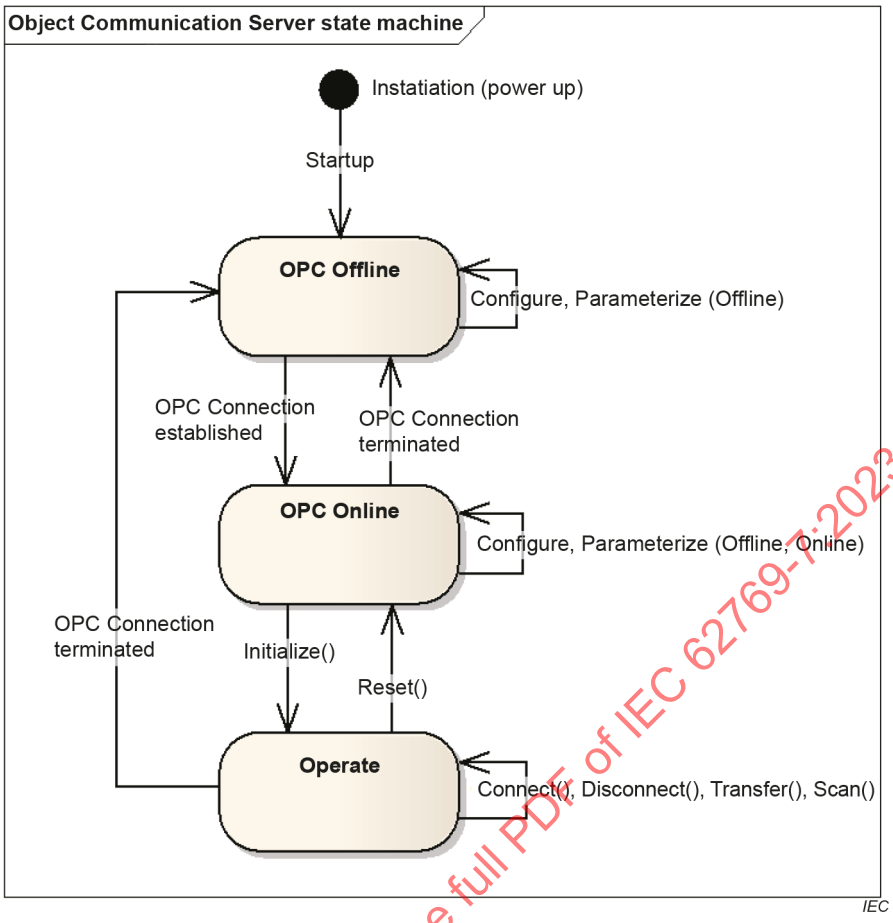


Figure 10 – FDI® Communication Server state machine

7.8.2 Deployment

The FDI® Server imports the FDI® Communication Package. The handling of EDD and UIP parts matches with the import procedure performed for the FDI® Package (see IEC 62769-2 and IEC 62769-3). The FDI® Communication Package represents the Communication Server Type.

The installation procedure described in the following is optional. (An embedded FDI® Communication Server need not provide an installation procedure.)

The FDI® Communication Server Installer (see 7.6) is a separate element. The installation procedure is started manually.

Depending on the operating systems, the execution of installation programs could require administration rights.

7.8.3 Server configuration

The FDI® Communication Server shall implement the means enabling the OPC UA Server specific configuration setting the link to the Discovery Server and the name of the FDI® Communication Server. According to 7.8.4, the FDI® Communication Server needs to know the address information about the Discovery Server. This document does not prescribe the way of how to do this.

The bootstrap process needed to establish the connection between an OPC UA Client and an OPC UA Server requires some administration work. A simple plug and play does not seem possible.

7.8.4 Start up

The following definitions about the Server discovery mechanism refer to the definitions found in IEC 62541-4.

The FDI® Communication Server executable shall be started according to one of the following described ways:

a) The FDI® Communication Server executable is loaded by means of the configured operating system function. If the FDI® Communication Server is installed on a hardware separated from the FDI® Server, the FDI® Communication Server executable shall be loaded by means of the configured operating system function (auto-start).

b) The FDI® Server invokes the FDI® Communication Server executable process. The related functions are specific to the operating system and the system vendor implementations.

The starting FDI® Communication Server process shall register itself at a Discovery Server using the service RegisterServer. This enables OPC UA Clients to obtain information about the connected FDI® Communication Server including the application description, existing endpoints and security information. The related OPC UA Services are FindServers and GetEndPoints. The Discovery Server is a process running outside the FDI® Communication Server.

After starting up, the FDI® Communication Server has the status "OPC Offline".

7.8.5 Shutdown

The following definitions about the Server discovery mechanism refer to the definitions found in IEC 62541-4.

The shutdown of the FDI® Communication Server process shall unregister itself at the Discovery Server using the service RegisterServer by setting argument isOnline value false.

7.8.6 Watchdog

The following definitions about the Server discovery mechanism refer to the definitions found in IEC 62541-4.

The FDI® Communication Server shall periodically use the service RegisterServer to state its ability to receive a connection from the FDI® Server. The frequency is 10 min. The FDI® Communication Server can envision a VARIABLE for configuration purposes.

7.8.7 Establish the OPC UA connection

The FDI® Server connects as an OPC UA Client to the FDI® Communication Server according to IEC 62541-4.

The FDI® Server (OPC UA Client) establishes a secure connection to the FDI® Communication Server (OPC UA Server) using the SecureChannel service set defined in IEC 62541-4. The functional principles are defined in IEC TR 62541-1.

The communication between the FDI® Server (OPC UA client) and the FDI® Communication Server (OPC UA server) shall be based on the OPC UA TCP transport protocol with the OPC UA Binary Encoding and OPC UA Secure Conversation.

After successfully establishing the OPC UA connection, the FDI® Communication Server enters the status "OPC Online".

7.8.8 Instantiate the Communication Server

The creation of a CommunicationServer instance in the FDI® Server hosted Information Model works the same way as the instantiation of a Device.

7.8.9 Configure the communication hardware

As described in 7.3.1, the FDI® Communication Server can support the configuration of modular communication hardware. The modular communication hardware configuration shall be performed via the services AddComponent and RemoveComponent (7.3.2.4, 7.3.2.5).

If the Action ValidateModules (see 5.2.9) is implemented, the FDI® Server shall invoke this method. If the Action result is OK, then the FDI® Server shall perform the synchronization using the sub device browse name matching and the invocation of the FDI® Communication Server hosted module management services.

If the FDI® Communication Server supports modular communication hardware configuration, the correct communication hardware configuration is a prerequisite for successful initialization as described in 7.8.12.

7.8.10 Configure the Network

The COMPONENT declaration defined in 5.2.2, 5.2.3, 5.2.4, and 5.2.7 enables a description based approach for the FDI® Server to configure the network connections. The FDI® Communication Server's ValidateNetwork Action (see 5.2.8) can be added to support the network topology validation function as described in 5.2.8. The network topology is only present in the FDI® Server hosted Information Model (see 7.5).

7.8.11 Parameterize

The FDI® Communication Server can require proper bus parameter adjustment prior to any communication service processing. The FDI® Communication Package contained user dialogs (UIP or UID) enable interactive bus parameter adjustment. The FDI® Communication Package can contain additional Business Logic for bus parameterization purposes. The editing of FDI® Communication Server parameters changes the content of the FDI® Server hosted Information Model. The FDI® Server shall perform synchronization using the parameters' browse name matching. The FDI® Server copies the modified values from the FDI® Server hosted Information Model to the FDI® Communication Server hosted Information Model.

A simple FDI® Communication Server shall provide all its configuration capabilities in its main EDD, not in the sub-modules used to expose the connection points. This allows FDI® Hosts not supporting modular devices to parameterize an FDI® Communication Server using the standard FDI® user interface mechanisms.

NOTE The FDI® Server can change Parameter values in arbitrary order.

7.8.12 Initialize

On invocation of the method Initialize (see 7.3.2.3), the FDI® Communication Server shall use the current parameter settings and communication hardware configuration for communication hardware initialization purposes.

After successful initialization, the FDI® Communication Server enters the status "Operate".

7.8.13 Create the communication service object

Prior to running any data exchange, at least one instance of type ServerCommunicationServiceType has to be present or created. One instance of ServerCommunicationServiceType shall be always present. Further instances of

ServerCommunicationServiceType can be created if the instance of ServerCommunicationDeviceType implements the method AddComponent.

7.8.14 Communication relation

The definitions of Clause 6 apply. The FDI® Communication Server specifics are defined in the subsequent text.

The state chart in Figure 11 describes the state flow of a single communication relation with added status changes that are related to OPC UA specifics. Beside the specific aspects, the definitions in Clause 6 apply as well.

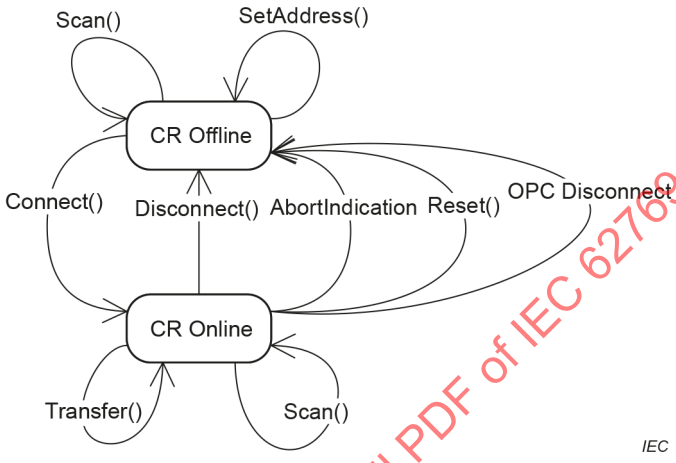


Figure 11 – Communication relation state chart

On invocation of the method Reset or the OPC Connection termination (OPC Connection loss), the FDI® Communication Server shall terminate all communication relations.

The FDI® Communication Server shall reject any parameterization or configuration change attempt in the status "Operate".

The FDI® Communication Server shall reject any communication relation related operation if the FDI® Communication Server status is different than "Operate".

If the Communication Server supports multiple instances of ServerCommunicationServiceType, these instances need to share information about existing communication relations.

7.8.15 Connect

Prior to running any information exchange related communication, the FDI® Communication Server requires establishing a communication relation between the device application and the physical network connected device. This happens through invocation of the method Connect.

The FDI® Communication Server shall be able to manage multiple communication relations. After successful execution of the method Connect, the corresponding communication relation enters the status "CR Online".

Because of the direct association of method Connect to a single communication service provider instance, the Communication Device knows the corresponding physical network connection.

7.8.16 Disconnect

Invocation of the method Disconnect terminates a communication relation, which inhibits further information exchange related communication with the physical network connected device.

After execution of the method Disconnect the corresponding communication relation enters the status "CR Offline". The communication relation becomes invalid.

7.8.17 Abort indication

Depending on protocol specifics, the FDI® Communication Server can detect communication aborts. Such communication abort indications are returned as communication service results during processing of the methods Transfer or Scan. After the FDI® Communication Server has returned an Abort Indication, the current communication relation enters the status "CR Offline". The communication relation becomes invalid.

7.8.18 Scan

The topology scan function can be invoked independently from an existing communication relation. Scan service details are specified in 7.3.3.2.

7.8.19 SetAddress

It depends on the protocol whether the address assignment service shall work even when a communication relation is already established.

8 FDI® Communication Gateway definition

8.1 General

A Gateway is a Communication Device that enables to bridge between different physical networks or different protocols. The logical representation of a Gateway device within the FDI® Server hosted Information Model enables the FDI® Server to process communication in heterogeneous network topologies.

8.2 Information Model

8.2.1 General

The Information Model of a Gateway is based on the Information Model defined in IEC 62769-5. Figure 12 replicates the Modular Device structure and its integration in the overall FDI® Server hosted Information Model.

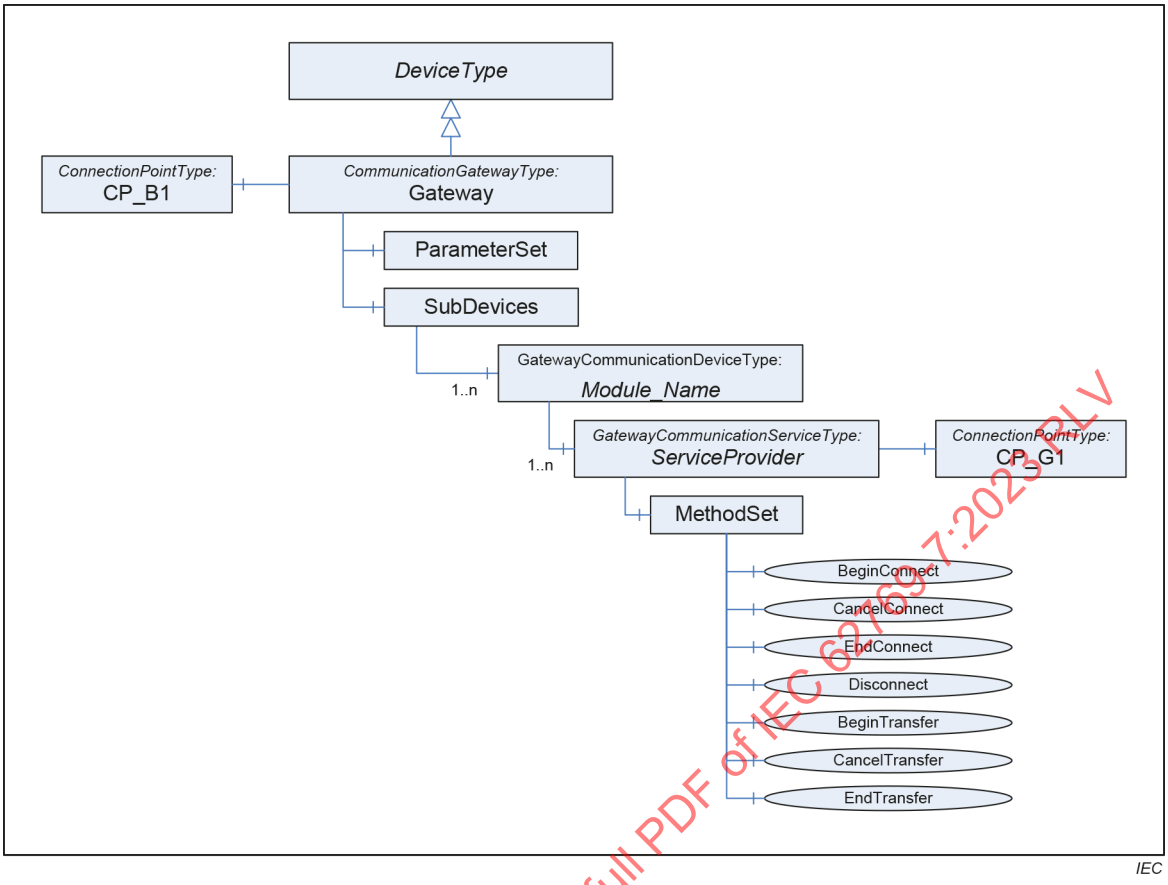


Figure 12 – Gateway information model

The Gateway is connected to the Network (see IEC 62769-5) through an instance of a ConnectionPointType (CP_B1). CP_B1 represents the FDI® Server assigned object (see IEC 62769-5) identification (name).

The Gateway is an instance of DeviceType. The optionally available ParameterSet (see IEC 62769-5) shall contain all device parameters that parameterize the communication interface used for Gateway initiated communication service requests.

The elements underneath SubDevices represent the physical or logical access to the network media of the Gateway. The attribute Module_Name represents the FDI® Server assigned object (see IEC 62769-5) identification (browse name).

The Gateway's communication service processing capabilities are accessible through multiple communication service provider instances created from GatewayCommunicationServiceType. The Business Logic behind the service methods implements the protocol translation function that is associated with the communication service interface.

NOTE Compared to the FDI® CommunicationServer, the Gateway does not support the transport of unsolicited messages, see 7.3.4.5.

8.2.2 CommunicationGatewayType

The CommunicationGatewayType is a subtype of the DeviceType. Figure 13 shows the CommunicationGatewayType definition. It is formally defined in Table 29.

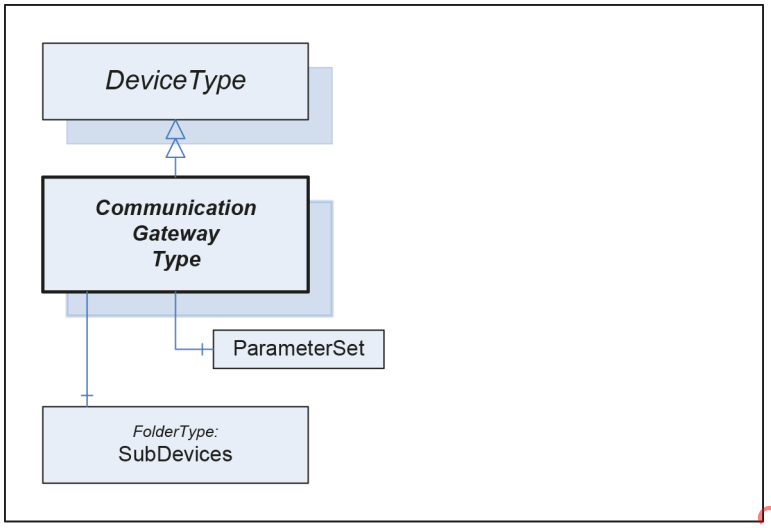


Figure 13 – CommunicationGatewayType

Table 29 – CommunicationGatewayType definition

Attribute	Value				
BrowseName	CommunicationGatewayType				
IsAbstract	False				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the DeviceType defined in IEC 62541-100.					
HasComponent	Object	SubDevices		FolderType	Mandatory
HasComponent	Object	ParameterSet			Optional

The module management needed to support a configurable communication hardware structure is based on the COMPONENT definitions for the entire CommunicationGatewayType. The COMPONENT definitions provide sufficient information to run generic module setup configuration.

NOTE The indication for a static communication hardware layout is present with the COMPONENT attribute CAN_DELETE set to FALSE for all COMPONENT declarations related to the sub devices of the entire device.

8.2.3 GatewayCommunicationDeviceType

8.2.3.1 General

The GatewayCommunicationDeviceType represents a communication module or channel connected to a particular network. The GatewayCommunicationDeviceType is a subtype of the DeviceType. The ParameterSet for each GatewayCommunicationDeviceType will contain Parameters necessary to configure the operation of the network. The protocol specific, mandatory bus parameters are specified in IEC 62769-4:2023, Annex F. Figure 14 shows the GatewayCommunicationDeviceType definition that is formally defined in Table 30 and Table 31.

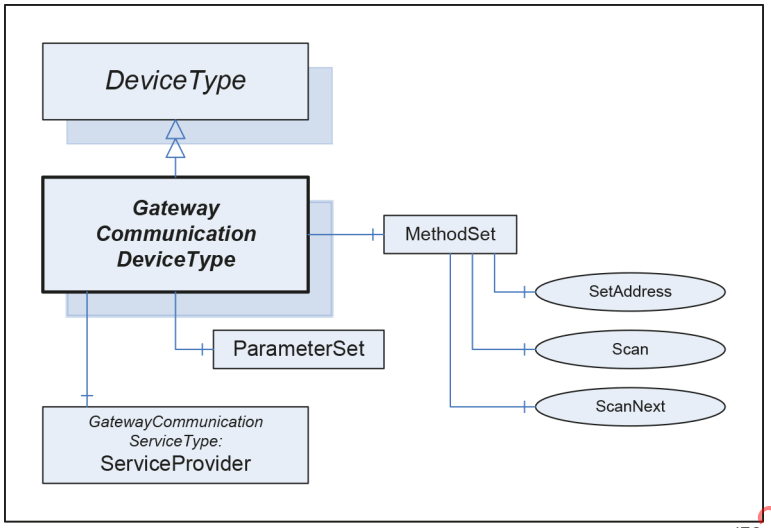


Figure 14 – GatewayCommunicationDeviceType

Table 30 – GatewayCommunicationDeviceType definition

Attribute	Value				
BrowseName	GatewayCommunicationDeviceType				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the DeviceType defined in OPC UA Part DI.					
HasComponent	Object	MethodSet		BaseObjectType	Mandatory
HasComponent	Object	ParameterSet		BaseObjectType	Optional
HasComponent	Object	ServiceProvider		Gateway Communication ServiceType	Mandatory

Table 31 – MethodSet of GatewayCommunicationDeviceType

Attribute	Value				
BrowseName	MethodSet				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
HasComponent	Method	Scan			Optional
HasComponent	Method	ScanNext			Optional
HasComponent	Method	SetAddress			Optional

8.2.3.2 Scan Method

Scan shall be used to start discovering physical network connected devices. The associations between the method Scan and the corresponding physical network connection enable the FDI® Communication Server to access the correct physical network connection. The Scan Method is implemented by an EDDL based method.

Because EDDL logic is not designed to handle XML documents the Scan service signature deviates from the definition given in 7.3.3.2. The scan service implementation returns the discovered devices using the LIST construct containing the COLLECTION instances. (Details are specified in 8.2.3.4.) The COLLECTION instances represent the Connection Point instances.

The signature of this Method is specified below. Table 32 and Table 33 specify the arguments and AddressSpace representation, respectively.

Signature

```
Scan([out] Int32 ServiceError)
```

Table 32 – Scan Method arguments

Argument	Description
ServiceError	0: OK 1: OK / get complete scan result by calling ScanNext -1: Failed / not initialized -2: Failed / not connected to a network -3: Failed / no device found the topologyScanResult is empty

Table 33 – Scan Method AddressSpace definition

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

8.2.3.3 ScanNext Method

ScanNext shall be used to continue discovering physical network connected devices. The associations between the method Scan and the corresponding physical network connection enable the FDI® Communication Server to access the correct physical network connection. The Scan Method is implemented by an EDDL based method.

Because EDDL logic is not designed to handle XML documents, the Scan service signature deviates from the definition given in 7.3.3.2. The scan service implementation returns the discovered devices using the LIST construct containing COLLECTION instances. (Details are specified in 8.2.3.4.) The COLLECTION instances represent the Connection Point instances.

The signature of this Method is specified below. Table 34 and Table 35 specify the arguments and AddressSpace representation, respectively.

Signature

```
ScanNext([out] Int32 ServiceError)
```

Table 34 – ScanNext Method arguments

Argument	Description
ServiceError	0: OK 1: OK / get complete scan result by recalling ScanNext -1: Failed / not initialized -2: Failed / not connected to a network -3: Failed / no device found the topologyScanResult is empty

Table 35 – ScanNext Method AddressSpace definition

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

8.2.3.4 Scan List

Each EDD describing the Connection Point of a Gateway shall describe the LIST element that holds the list of discovered devices after the successful execution of the Scan service.

```
LIST <Id>
{
    TYPE <ListEntry>;
}
```

<ID>: The identifier shall match with the SCAN _LIST attribute value described in 8.3.2.3.

<ListEntry>: The attribute value shall refer to the COLLECTION definition that describes the Connection Point as defined in 8.3.2.5.

8.2.3.5 SetAddress Method

For definitions, see 7.3.3.5.

8.2.4 GatewayCommunicationServiceType

8.2.4.1 General

Communication services provide the means to communicate with a Device or to e.g. execute a Scan on some Network. Communication services are represented through EDDL based methods (business logic) provided with the FDI® Device Package contained EDD.

The implementation of all communication services except the Disconnect service follows an asynchronous pattern. This implies that each communication service is split into three methods. This allows the FDI® Server to execute communication services in parallel as well as cancel operations that last too long.

Begin<name>: This method starts the execution of the service. The method returns either the execution state of <name> or the result if it is immediately present.

End<name>: This method checks if the result of the service is already available that was started using a preceding Begin<name> call. Like Begin<name> this method returns either the execution state or the result if available.

Cancel<name>: This method cancels a started service execution.

A service identification number (ServiceId) enables the Communication Gateway to keep track of the relation between the method calls that belong to a single communication service process. If the Communication Gateway supports multiple instances of GatewayCommunicationServiceType these instances do share information about currently used ServiceIds. Thus, a communication service has to be executed on a single instance of GatewayCommunicationServiceType.

To reduce unnecessary poll cycles, both methods Begin<name> End<name> return a delay time value (DelayForNextCall). The method caller shall delay the invocation of the method End<name> according to the returned argument value.

Figure 15 shows the GatewayCommunicationServiceType definition that is formally defined in Table 36 and Table 37.

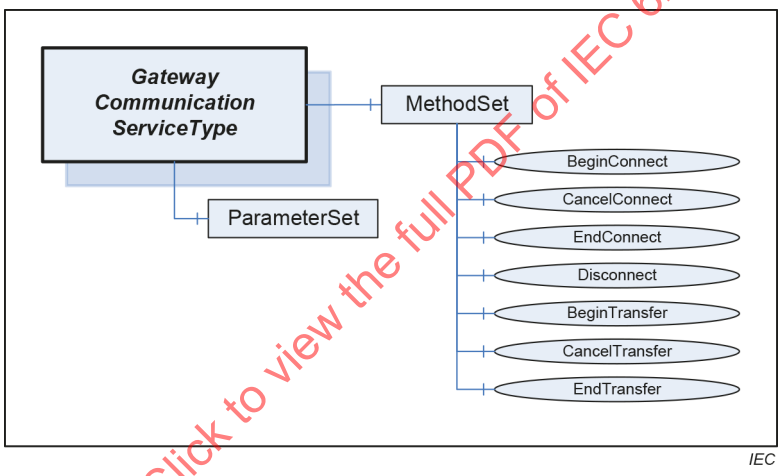


Figure 15 – GatewayCommunicationServiceType

Table 36 – GatewayCommunicationServiceType definition

Attribute	Value				
BrowseName	GatewayCommunicationServiceType				
IsAbstract					
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Subtype of the DeviceType defined in OPC UA Part DI.					
HasComponent	Object	MethodSet		BaseObjectType	Mandatory
HasComponent	Object	ParameterSet		BaseObjectType	Optional

Table 37 – MethodSet of GatewayCommunicationServiceType

Attribute	Value				
BrowseName	MethodSet				
IsAbstract					
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
HasComponent	Method	BeginConnect			Optional
HasComponent	Method	CancelConnect			Optional
HasComponent	Method	EndConnect			Optional
HasComponent	Method	Disconnect			Optional
HasComponent	Method	BeginTransfer			Mandatory
HasComponent	Method	CancelTransfer			Mandatory
HasComponent	Method	EndTransfer			Mandatory

8.2.4.2 Connect service

The Connect service shall be used to establish a communication relation to a device that is physically connected to the Network. Establishing the communication relation can imply checks of identification data that are part of the addressData with data inside the physical device. The service performs this DeviceType match verification according to a corresponding network protocol standard. Related details are specified in IEC 62769-4:2023, Annex F.

The devices address is contained in the ConnectionPoint of the corresponding Device Instance within the Information Model (Device Connection Point). The communication relation between the Information Model associated device application and the physical device is further on identified by the communication relation identifier. Details about how to manage the status of a communication relation are described in Clause 6.

NOTE 1 As the NodeId is a unique identifier within the Information Model scope, the NodeId of the Device Connection Point can be a unique identifier for any communication relation in the scope of a Communication Device.

NOTE 2 The term communication relation is introduced to describe the status of an infrastructure that enables data exchange between the information model hosted data and a physical device. If the communication relation is established, data exchange is possible.

The signatures of the Connect service methods are specified below. Table 38 specifies the arguments.

Signature

```
BeginConnect(
   [in]   ByteString       CommunicationRelationId,
   [in]                    <AddressData>,
   [out]                   <DeviceInformation>,
   [in]   UInt32           ServiceID,
   [out]  UInt32           DelayForNextCall,
   [out]  Int32            ServiceError);

EndConnect(
   [in]   ByteString       CommunicationRelationId,
   [out]                   <DeviceInformation>,
   [in]   UInt32           ServiceID,
   [out]  Uint32           DelayForNextCall,
   [out]  Int32            ServiceError);

CancelConnect(
   [in]   ByteString       CommunicationRelationId,
   [in]   Uint32           ServiceID,
   [out]  Int32            ServiceError);
```

Table 38 – Connect Method arguments

Argument	Description
CommunicationRelationId	This is a client generated id that is used to uniquely identify this connection. This could be an index (e.g., a NodeId) that the client (= FDI® Server) needs to identify entries in its topology.
<AddressData>	This is a placeholder for a protocol specific argument list that is used for the address and optional device identification data (details are described in IEC 62769-4:2023, Annex F.
<DeviceInformation>	A protocol specific argument list in which the connect routine stores the resulting data.
ServiceId	The service transaction code establishes the relation between the service request and the corresponding response.
DelayForNextCall	The value specifies a delay time in ms, the caller shall wait before the next invocation of EndConnect.
ServiceError	0: OK / function started asynchronously, result has to be polled with EndConnect 1: OK / execution finished, connection established successfully -1: Connect Failed / cancelled by caller -2: Call Failed / unknown service ID -3: Connect Failed / device not found -4: Connect Failed / invalid device address -5: Connect Failed / invalid device identification

8.2.4.3 Disconnect method

Specified in 7.3.4.3.

8.2.4.4 Transfer service

Transfer shall be used to perform information exchange with a Device.

The signatures of the Transfer service methods are specified below. All arguments are specified in Table 39.

Signature

```
BeginTransfer(
   [in]   ByteString          CommunicationRelationId,
   [in]                             <SendData>,
   [out]                            <ReceiveData>,
   [in]   UInt32              ServiceId,
   [out]  UInt32              DelayForNextCall,
   [out]  Int32                     ServiceError);

EndTransfer(
   [in]   ByteString          CommunicationRelationId,
   [out]                            <ReceiveData>,
   [in]   UInt32              ServiceId,
   [out]  UInt32              DelayForNextCall,
   [out]  Int32                     ServiceError);

CancelTransfer(
   [in]   ByteString          CommunicationRelationId,
   [in]   UInt32              ServiceId,
   [out]  Int32                     ServiceError);
```

Table 39 – Transfer Method arguments

Argument	Description
CommunicationRelationId	See 8.2.4.2.
<SendData>	This is a placeholder for a protocol specific list of arguments as described in IEC 62769-4:2023, Annex F. The argument values represent the protocol specific communication service request that is sent to the device.
<ReceiveData>	This is a placeholder for a protocol specific list of arguments as described in IEC 62769-4:2023, Annex F. The argument values represent the protocol specific communication service response that is received from the device.
ServiceId	The service transaction code establishes the relation between the service request and the corresponding response.
DelayForNextCall	The value specifies a delay time in ms, the caller shall wait before the next invocation of EndTransfer.
ServiceError	0: OK / function started asynchronously, result has to be polled with EndTransfer 1: OK / execution finished, ReceivedData contains the result -1: Transfer Failed / cancelled by caller -2: Call Failed / unknown service ID -3: Transfer Failed / no existing communication relation -4: Transfer Failed / invalid communication relation identifier -5: Transfer Failed / invalid sendData content -6: Transfer Failed / invalid receiveData format

8.3 FDI® Communication Package

8.3.1 General

Subclause 8.3 specifies the FDI® Communication Package details that are specific for Gateways. The definitions given in 5.1 apply also.

8.3.2 EDD

8.3.2.1 General

The definitions in 5.2 apply. Additionally, the EDD elements as specified in IEC 62769-4 and provided with a Gateway specific FDI® Communication Package shall contain:

a) a PROFILE (IEC 61804-3): The PROFILE definition shall be chosen according to the protocol used for communication service requests;

b) a Business Logic: The communication service provider related EDD COMPONENTs shall implement the methods specified in IEC 61804-3. These methods implement the protocol bridging logic. The translation procedures open out into outbound communication requests via the EDDL Builtin library function invocation or the writing of data onto an online node. The set of usable EDDL Builtin library functions is bound to the PROFILE;

c) a Module management: The Gateway related Component can implement the ValidateModules (see 5.2.9). The implemented logic shall validate individual changes and handle any parameter related dependencies for the whole Gateway device. The implementation of ValidateModules is optional when the actual product specific COMPONENT declaration is sufficient to configure the module setup without any additional Business Logic.

8.3.2.2 Gateway Component

Each FDI® Communication Package describing a Gateway shall contain an EDD element describing the Gateway as defined in 5.2.2. Gateway specifics are described in the following:

```
COMPONENT <DeviceComponentId>
{
     LABEL "<Label>";
     CAN_DELETE TRUE;
     CHECK_CONFIGURATION <ValidateModules>;
     CLASSIFICATION NETWORK_COMPONENT;
     COMPONENT_RELATIONS
     {
        <CommunicationDeviceRelationId>
     }
     PROTOCOL <ProtocolId>;
}
```

<ProtocolID>: The existence of this attribute indicates the connectivity of the Gateway regarding outbound communication. It allows the FDI® Server to find the network using the same type of protocol to which this Gateway can be connected. For standardized protocols, the value is defined by the related fieldbus organization.

8.3.2.3 Gateway CommunicationDevice Component

Each FDI® Communication Package describing a Gateway shall contain at least one EDD element describing at least one CommunicationDevice component as defined in 5.2.3.

NOTE A Gateway is sometimes referred to as "Remote IO". Remote IO is a Modular Device supporting multiple various module types that can be flexibly assigned to any slot. Thus it is possible to create multiple different Remote IO configurations (n – slots X m – module types).

The rules about COMPONENT attribute settings shown in 5.2.3 apply. Gateway specifics are described in the following:

```
COMPONENT <CommunicationDeviceComponentId>
{
    LABEL "<Label>";
    CAN_DELETE <CanDelete>;
    CLASSIFICATION NETWORK_COMPONENT;
    COMPONENT_RELATIONS
    {
<CommunicationServiceProviderRelationId>
    }
    SCAN <Scan>;
    SCAN_LIST <ScanList>;
}
```

<Scan>: The attribute refers to the METHOD implementing the device discovery service. The reference works because the identifier value of the METHOD matches with attribute value. (Implementation details are specified in 8.2.3.2.)

<ScanList>: The attribute value refers to the LIST describing the topology scan results. This list shall contain all devices discovered during execution of the device discovery service. (Implementation details are specified in 8.2.3.4.)

8.3.2.4 Communication Service component

The rules defined in 5.2.4 apply.

Additionally, the file describing the component contains the implementations of the methods defined in 8.2.4.2, 8.2.4.3 and 8.2.4.4. The identifier used for the related METHOD constructs matches with the method names specified in Table 37.

8.3.2.5 Connection Point Component

The rules defined in 5.2.5 apply.

8.3.2.6 Connection Point Collection

The rules defined in 5.2.6 apply.

8.4 Handling and behaviour

8.4.1 General

A Gateway provides functionality to communicate between two communication protocols. Gateways are used to communicate from one network of the automation system to another (subordinated) network. Figure 16 shows a typical example where a HART®[2] device is connected to a PROFIBUS®[3] Remote I/O. In order to communicate to the HART device, the Remote I/O receives a communication request via the PROFIBUS network (see Figure 16, key 1). The communication request contains the necessary information that allows the gateway to create the according HART Command and send it to the HART device (see Figure 16, key 2).

---

2 HART® is a registered trademark of the non-profit organization FieldComm Group, Inc. This information is given for convenience of users of this document and does not constitute an endorsement by IEC of the trademark holder or any of its products. Compliance does not require use of the trademark. Use of the trademark requires permission of the trademark holder.

3 PROFIBUS® is the registered trademark of the non-profit organization PROFIBUS Nutzerorganisation e.V. (PNO). This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the trademark holder or any of its products. Compliance does not require use of the trademark. Use of the trademark requires permission of the trademark holder.

The way the HART Command is wrapped into the PROFIBUS communication request can be standard or gateway specific. The HART response (see Figure 16, key 3) from the device is embedded as a PROFIBUS communication response (see Figure 16, key 4). The way the Gateway wraps the response can either be standard or Gateway specific.

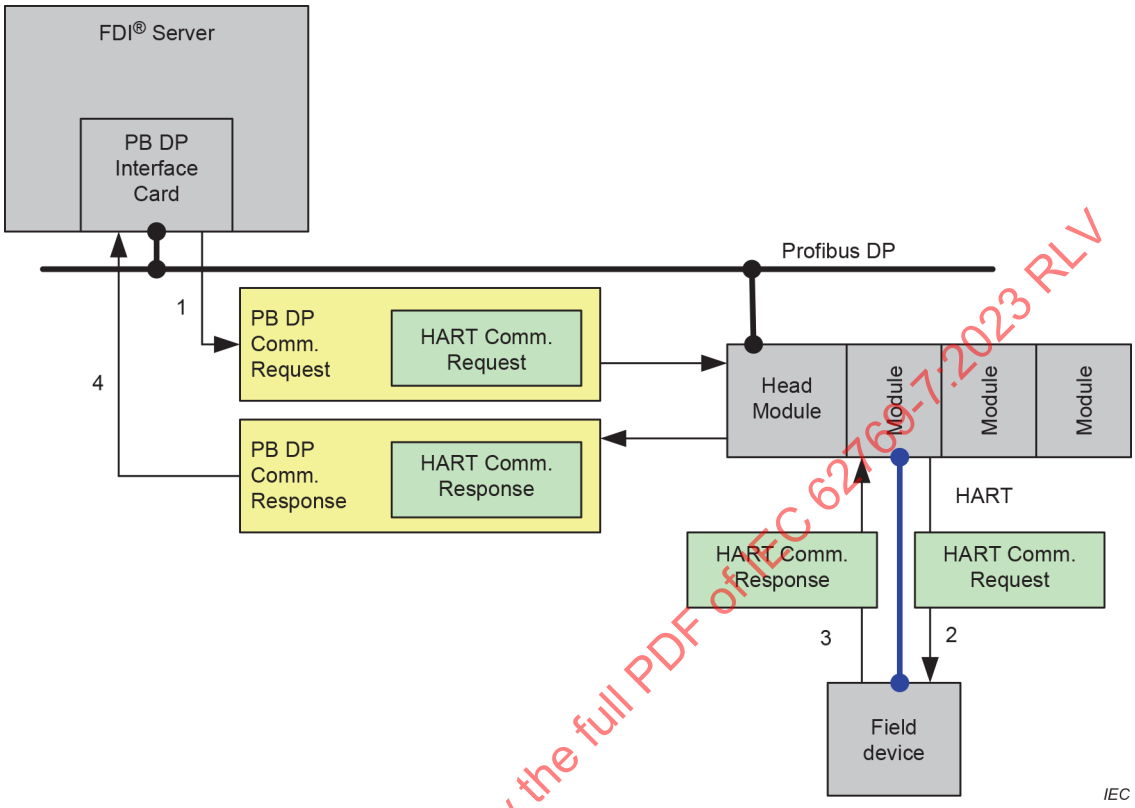


Figure 16 – Nested Communication

Subclause 8.4 defines the Gateway handling and behaviour rules along the life cycle beginning with the deployment, start up, bus commissioning, until the communication services processing.

8.4.2 Deployment

The definitions of 5.2.11 apply.

8.4.3 Start up

The Information Model and the FDI® Package EDD element based Gateway representation does not require any start up procedures.

8.4.4 Configure the communication hardware

The handling and behaviour matches with the specification in 7.8.9. The only difference is in the channel setup that is represented in the FDI® Server hosted Information Model only.

8.4.5 Configure the Network

The handling and behaviour matches with the specification in 7.8.10.

8.4.6 Parameterize

The Gateway can require proper parameter adjustment prior to any communication service processing. The FDI® Communication Package contained user dialogs (UID or UIP) enable

interactive bus parameter adjustment. The FDI® Communication Package can provide additional Business Logic for bus parameterization purposes.

8.4.7 Communication relation

The status machine definitions in Clause 6 apply.

If the Communication Gateway supports multiple instances of GatewayCommunicationServiceType, these instances need to share information about existing communication relations.

8.4.8 Connect

Prior to running any information exchange related communication, the Gateway requires establishing a communication relation between the device application and the physical network connected device. This happens through invocation of the method Connect. This enables the Gateway to perform an optional outbound communication service request, which might be needed for a specific Gateway device to establish a communication relation.

The Gateway shall be able to manage multiple communication relations. After successful execution of the method Connect, the corresponding communication relation enters the status "CR Online".

Invocation of the services Transfer and Scan is allowed in the status "Online" only.

8.4.9 Disconnect

Invocation of the method Disconnect terminates a communication relation, which inhibits further information exchange related communication with the physical network connected device.

After execution of the method Disconnect the corresponding communication relation enters the status "CR Offline". The communication relation becomes invalid.

8.4.10 Abort indication

Depending on protocol specifics, the Gateway can detect communication aborts. Such communication abort indications are returned as communication service results during the processing of the methods Transfer or Scan. After the Gateway has returned an Abort Indication, the current communication relation enters the status "CR Offline". The communication relation becomes invalid.

8.4.11 Scan

Gateways declare their device discovery service capability based on the EDDL construct COMPONENT – the attributes SCAN and SCAN_LIST. Related definitions are defined in IEC 61804-3. The SCAN attribute setting is mandatory within the EDD element describing the Gateway CommunicationDevice. The attribute value refers to the METHOD executing the topology scan function. The SCAN_LIST attribute setting is mandatory within the Gateway CommunicationDevice COMPONENT. The attribute value refers to the element that contains the topology scan result created by the method referenced by the attribute value SCAN. The SCAN_LIST shall refer to a LIST containing COLLECTION elements describing the detected devices (Scan-List-Item). The protocol specific content of a Scan-List-Item is described in IEC 62769-4:2023, Annex F.

The invocation of the functions Scan and ScanNext results in outbound communication service requests.

8.4.12 Communication Error Handling

The communication service processing shall use the EDDL Builtin function provided abort according to the EDDL Profiles:

- during the creation of communication messages,
- during the processing of the response from the communication request.

The communication service processing does not trigger communication errors based on communication timeouts.

Annex A
(informative)

Layered protocols

A.1 General

Ethernet based protocols commonly consist of a stack of different protocols based on the ISO/OSI model. Looking at the growing number of Ethernet based fieldbus protocols, it is crucial to have a common layered modeling concept for Connection Points based on the ISO/OSI model also.

Connection Points are the elements that contain the address information accessed by the Communication Devices to collect the information needed for communication. The semantics of Connection Point attributes need to be standardized.

The PROFINET®[4] device can concurrently support PROFINET, SNMP and HTTP. The information stored in the Connection Point of a device is different for each protocol due to different application layers. The information for layer 1 to 4 shall consistently hold the same information. Therefore, Connection Points shall inherit Connection Point information from lower network layers.

The problem is about how to ensure that address information from lower layers is named in a consistent way throughout all protocols that are built upon lower layers.

A.2 Convention for protocol specific annex creation

A.2.1 General

Since Connection Point description is based on EDDL, an actual inheritance approach known from object-oriented programming languages does not seem applicable. The approach described in Clause A.2 is based more on conventions. The naming convention shall ensure that the address value attribute names defined for a "lower level" protocol are reused in the Connection Point definitions for higher level protocols. The same holds true for the COLLECTION referred VARIABLES. VARIABLE declarations for higher-level protocols shall be copies from the VARIABLE declarations for lower-level protocols. The convention described here is applicable for the creation of protocol specific annexes.

A.2.2 Connection Point

The following shows some EDDL examples of how Connection Point declarations follow this naming convention.

```
COLLECTION ConnectionPoint_MAC
{
    LABEL "<Label>";
    MEMBERS
    {
        MAC,
        VALID
    }
```

[4] PROFINET® is the registered trademark of the non-profit organization PROFIBUS Nutzerorganisation e.V. (PNO). This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the trademark holder or any of its products. Compliance does not require use of the trademark. Use of the trademark requires permission of the trademark holder.

```
}



COLLECTION ConnectionPoint_IPv4
{
    LABEL "<Label>";
    MEMBERS
    {
        MAC,
        IPv4,
        VALID
    }
}



COLLECTION ConnectionPoint_TCPUDP
{
    LABEL "<Label>";
    MEMBERS
    {
        MAC,
        IP,
        PORT,
        VALID
    }
}



COLLECTION ConnectionPoint_PROFINET
{
    LABEL "<Label>";
    MEMBERS
    {
        MAC,
        IP,
        PORT,
        DNSNAME,
        VALID
    }
}


COLLECTION ConnectionPoint_HTTP
{
    LABEL "<Label>";
    MEMBERS
    {
        MAC,
        IP,
        PORT,
        URL,
        VALID
    }
}
```

A.3 FDI® Communication Package definition

A.3.1 Communication services

The actual communication service is always implemented according to the specific protocol. This is reflected by the different semantic of the communication service arguments specified by the protocol specific annexes. So the actual implementation of a ServerCommunicationDeviceType or GatewayCommunicationDeviceType can support just one protocol. Thus, if an FDI® Communication Server or Gateway is able to support multiple different protocols, it shall describe separate GatewayCommunicationDeviceTypes or ServerCommunicationDeviceTypes. The need to define protocol specific service sets represents the demand to separate the Connection Point and Network related COMPONENT definitions described in A.3.2 and A.3.3.

A.3.2 Connection Point

The FDI® Communication Package shall contain separate Connection Point descriptions for each supported protocol.

A.3.3 Network

The relation between the COMPONENT describing the network and the COMPONENT describing the Connection Point enables generic communication path detection and generic topology configuration. Thus the FDI® Communication Package shall contain a separate COMPONENT definition for each supported Network (protocol).

A.4 Representation in the Information Model

Connection Points sharing a certain set of address formation can contain redundant address information, for example the IP address is the same for the SNMP and PROFINET I/O.

If a Device and an FDI® Communication Server share a set of protocols, then that Device and FDI® Communication Server are associated through multiple separate networks.

A Device supporting multiple protocols can be connected to different FDI® Communication Devices that support only one protocol.

Annex B
(normative)

Namespace and Mappings

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

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

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

The BrowsePath for an Instance Node is constructed by appending the BrowseName of the Instance Node to the BrowseName for the containing instance or type. An underscore character is used to separate each BrowseName in the path.

The NamespaceUri http://FDI-cooperation.com/OpcUa/FDI7/ is applied to NodeIds defined here.

The CSV released with this version of the document is provided by FieldComm Group. An electronic version of the complete Information Model defined in this document is also provided. It follows the XML Information Model Schema syntax defined in IEC 62541-6.

The Information Model Schema released with this version of the document is provided by FieldComm, Inc Group.

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

INTÉGRATION DES APPAREILS DE TERRAIN (FDI®) –

Partie 7: Appareils de Communication

AVANT-PROPOS

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

L'IEC 62769-7 a été établie par le sous-comité 65E: Les dispositifs et leur intégration dans les systèmes de l'entreprise, du comité d'études 65 de l'IEC: Mesure, commande et automation dans les processus industriels. Il s'agit d'une Norme internationale.

Cette troisième édition annule et remplace la deuxième édition parue en 2021. Cette édition constitue une révision technique.

Cette édition inclut les modifications techniques majeures suivantes par rapport à l'édition précédente:

a) ajout de la Méthode ScanExtended.

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

Projet	Rapport de vote
65E/859/CDV	65E/916/RVC

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

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

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

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

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

- reconduit,
- supprimé,
- remplacé par une édition révisée, ou
- amendé.

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

INTÉGRATION DES APPAREILS DE TERRAIN (FDI®) –

Partie 7: Appareils de Communication

1 Domaine d'application

La présente partie de l'IEC 62769 spécifie les éléments de mise en œuvre des fonctions de communication, appelés Appareils de Communication.

L'architecture FDI®[1] complète est représentée à la Figure 1. Les composants architecturaux qui relèvent du domaine d'application du présent document ont été mis en évidence dans cette représentation. Le domaine d'application du document relatif aux Paquetages FDI® est limité aux Appareils de Communication. Le Serveur de Communication représenté à la Figure 1 est un exemple d'Appareil de Communication spécifique.

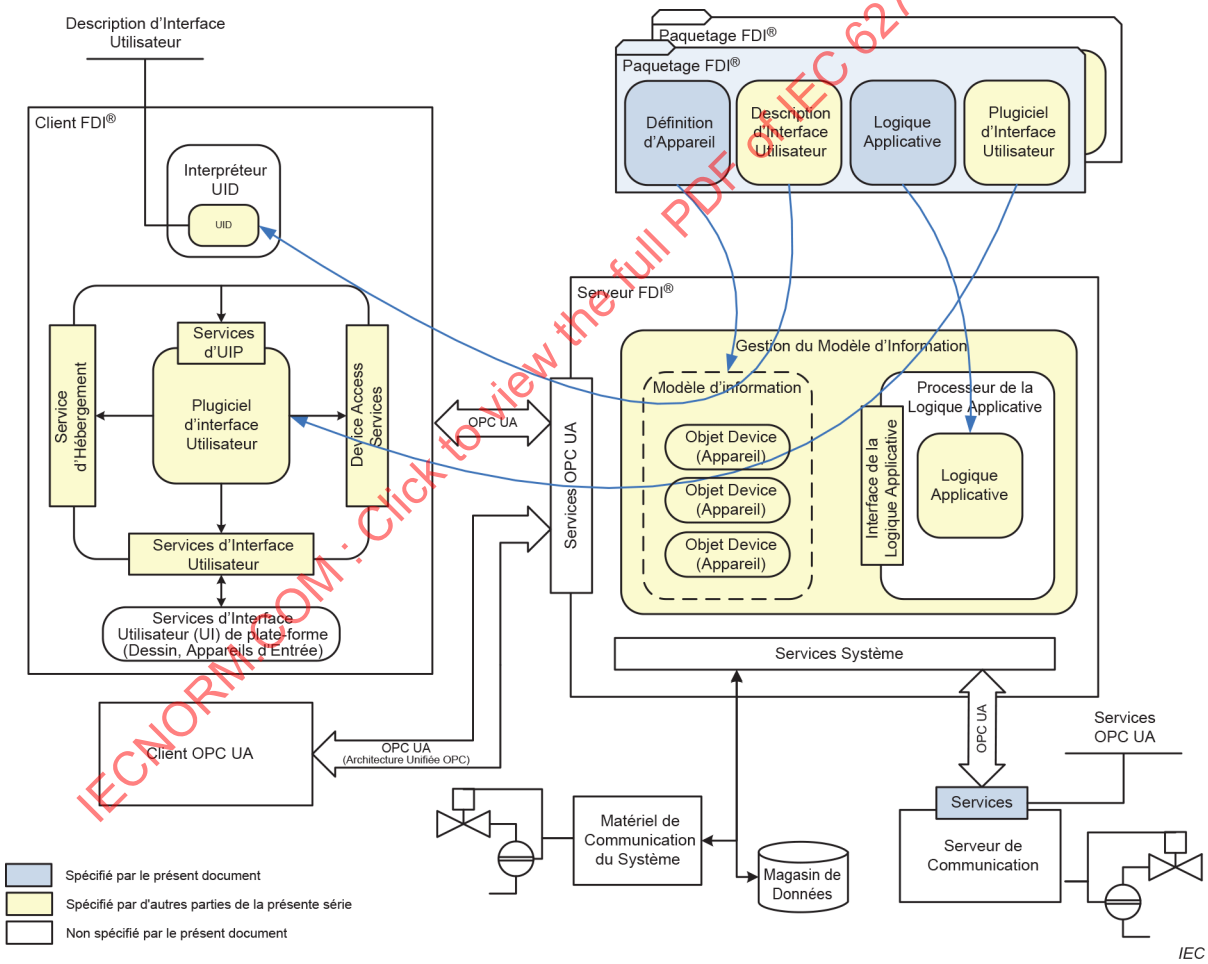


Figure 1 – Diagramme de l'architecture FDI®

---

1 FDI® est une marque déposée de l'organisation à but non lucratif Fieldbus Foundation, Inc. Cette information est donnée à l'intention des utilisateurs du présent document et ne signifie nullement que l'IEC approuve le détenteur de la marque ou l'emploi de ses produits. La conformité n'exige pas l'utilisation de la marque. L'utilisation de la marque exige l'autorisation du détenteur de la marque.

2 Références normatives

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

IEC 61804-3, *Les dispositifs et leur intégration dans les systèmes de l'entreprise – Blocs fonctionnels (FB) pour les procédés industriels et le langage de description électronique de produit (EDDL) – Partie 3: Sémantique et syntaxe EDDL*

IEC 61804-4, *Les dispositifs et leur intégration dans les systèmes de l'entreprise – Blocs fonctionnels (FB) pour les procédés industriels et le langage de description électronique de produit (EDDL) – Partie 4: Interprétation EDD*

IEC TR 62541-1, *OPC Unified Architecture – Part 1: Overview and concepts* (disponible en anglais seulement)

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

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

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

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

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

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

IEC 62769-3, *Intégration des appareils de terrain (FDI®) – Partie 3: Serveur*

IEC 62769-4:2023, *Intégration des appareils de terrain (FDI®) – Partie 4: Paquetages FDI®*

IEC 62769-5, *Intégration des appareils de terrain (FDI®) – Partie 5: Modèle d'Information FDI®*

3 Termes, définitions, abréviations, acronymes et conventions

3.1 Termes et définitions

Pour les besoins du présent document, les termes et définitions de l'IEC 62769-1, de l'IEC 62769-3 et de l'IEC 62541-6 ainsi que les suivants s'appliquent.

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

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

3.1.1
Passerelle
Appareil de Communication qui permet de relier différents réseaux physiques ou différents protocoles

3.2 Abréviations et acronymes

Pour les besoins du présent document, les abréviations et acronymes de l'IEC 62769-1, de l'IEC 62541-6 ainsi que les suivants s'appliquent.

HTTP (Hypertext Transfer Protocol)	Protocole de transport hypertexte
IP (Internet Protocol)	Protocole Internet
PHY	Matériel de communication physique
SNMP (Simple Network Management Protocol)	Protocole de gestion de réseau simple
TCP (Transmission Control Protocol)	Protocole de contrôle de transmission
URI (Uniform Resource Identifier)	Identificateur uniforme de ressource

3.3 Conventions

3.3.1 Syntaxe EDDL

La présente partie de l'IEC 62769 spécifie le contenu du composant EDD qui fait partie des Paquetages de Communication FDI®. Le contenu de la spécification qui utilise la syntaxe EDDL est rédigé avec la police `Courier New`. La syntaxe EDDL est utilisée pour les déclarations de signature de méthode, de variable, de structure de données et de composant.

3.3.2 Utilisation de majuscules

La mise en majuscules de la première lettre des mots est utilisée dans la série IEC 62769 pour souligner un terme défini spécifique à la FDI®.

3.3.3 Notation graphique

Le présent document utilise la notation graphique définie dans l'IEC 62769-5.

4 Vue d'ensemble

Le terme abstrait Appareil de Communication FDI® représente une entité qui met en œuvre des fonctions de communication sur un réseau, en utilisant un protocole spécifique. Le groupe des Appareils de Communication FDI® se divise en deux groupes principaux.

a) Le Serveur de Communication FDI® est un Serveur dédié de l'architecture unifiée OPC (OPC UA) qui fournit l'accès à un ou plusieurs réseaux d'appareils de terrain. Le Serveur de Communication FDI® est spécifié à l'Article 7.

b) La Passerelle de Communication FDI® permet de relier différents réseaux physiques ou différents protocoles. La logique applicative du pontage est mise en œuvre dans le composant EDD qui est fourni avec un Paquetage de Communication FDI®. La Passerelle de Communication FDI® est spécifiée à l'Article 8.

NOTE Les principales différences entre une Passerelle et un Serveur de communication sont les suivantes: En matière de FDI®, le Serveur de Communication FDI® est un Serveur OPC UA dédié qui fournit l'accès à un ou plusieurs réseaux d'appareils de terrain. Une Passerelle est un Appareil de Communication qui permet de relier différents réseaux physiques ou différents protocoles. La représentation logique d'une passerelle dans le modèle d'information hébergé par le Serveur FDI® permet au serveur FDI® Server de traiter la communication dans des topologies de réseaux hétérogènes.

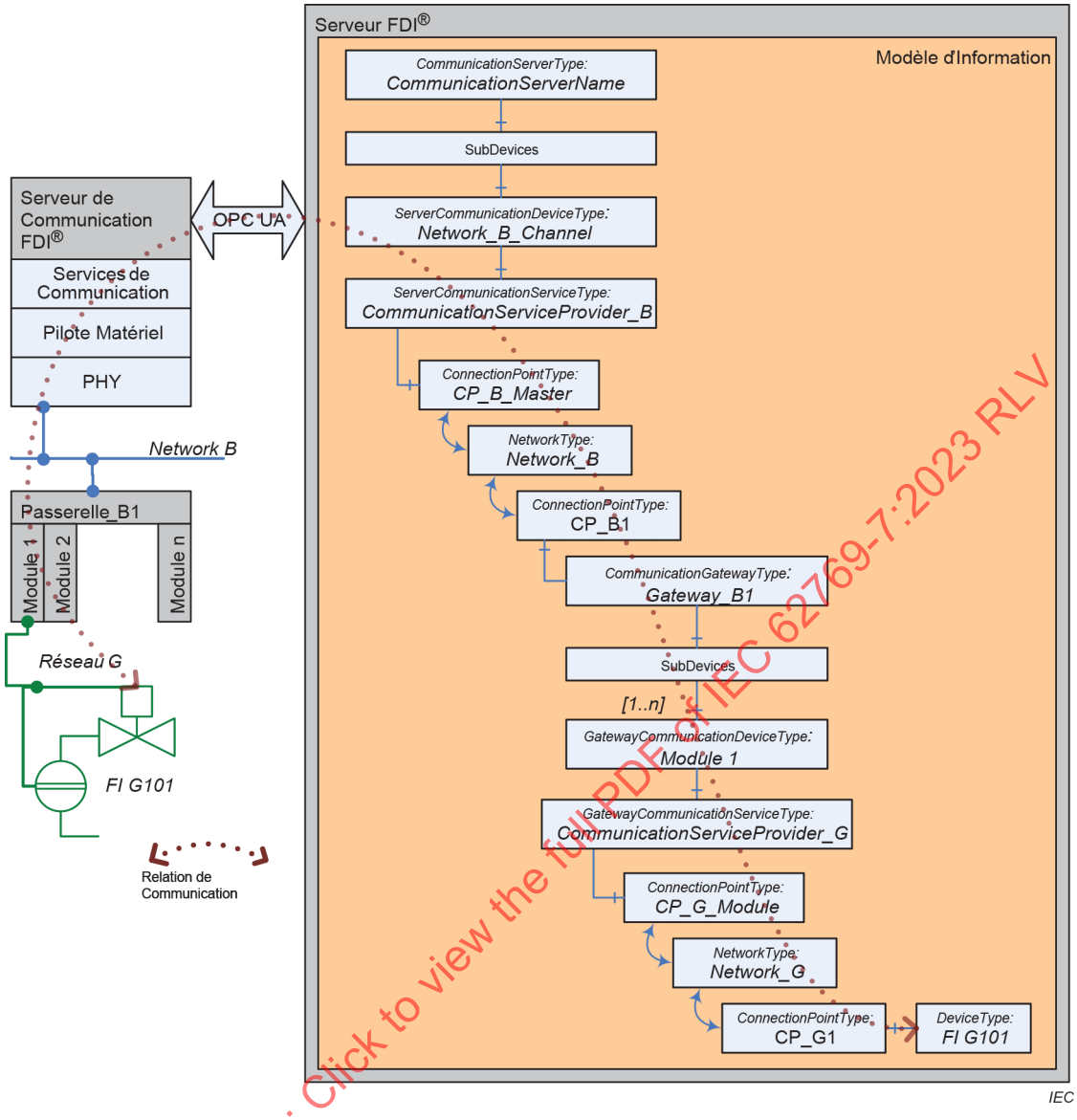


Figure 2 – Architecture de l'infrastructure de communication FDI®

Le Modèle d'information hébergé par le Serveur FDI® contient une représentation de la topologie du réseau (voir également IEC 62769-5). Le Modèle d'information représenté à la Figure 2 est un exemple qui montre comment les éléments utilisés par le Modèle d'information reflètent la topologie réelle du réseau.

1) L'instance de CommunicationServerType (dénommée CommunicationServerName) représente le Serveur de Communication FDI®. Le Serveur de Communication FDI® met en œuvre l'accès au réseau de communication physique (matériel de communication). L'Article 7 décrit les éléments spécifiques du Modèle d'Information, le contenu exigé pour le Paquetage de Communication FDI® et le traitement des éléments qui s'y trouvent. (En ce qui concerne les sous-appareils, voir l'IEC 62769-5).

2) L'instance de ServerCommunicationDeviceType et de ServerCommunication-ServiceType (dénommée Network_B_Channel) est mappée aux services de communication mis en œuvre par le Serveur de Communication FDI®. Le ServerCommunicationDeviceType est spécifié en 7.3.3. Le ServerCommunicationServiceType est spécifié en 7.3.4.

3) L'instance de CommunicationGatewayType (dénommée Gateway_B1) représente la Passerelle physique. L'Article 8 décrit les éléments spécifiques associés du Modèle d'Information, le contenu exigé pour le Paquetage FDI® et le traitement des éléments qui s'y trouvent.

4) L'instance de GatewayCommunicationDeviceType (dénommée Module 1) est mappée à un module physique ou logique, ce qui permet la communication au réseau auquel ce module est connecté. Le GatewayCommunicationDeviceType est spécifié en 8.3.2.3. Les éléments spécifiques associés de la Passerelle sont décrits à l'Article 8.

5) L'instance de GatewayCommunicationServiceType (dénommée CommunicationServiceProvider_G) représente la capacité des Passerelles à traiter les services de communication. La mise en œuvre spécifique à la Passerelle de GatewayCommunicationServiceType est fondée sur la Logique Applicative qui permet d'exécuter les services de communication dans des réseaux de communication hétérogènes.

6) Une relation de communication (de plus amples informations sont données à l'Article 6) entre un appareil physique et la représentation de l'appareil gérée par le Serveur FDI® est toujours associée aux objets du service de communication qui sont des instances d'un GatewayCommunicationServiceType ou d'un ServerCommunicationServiceType. La capacité d'instanciation de plusieurs protocoles de supports d'objets du service de communication permet de traiter plusieurs connexions logiques entre un maître bus et un appareil.

7) Le Modèle d'Information représente les connexions entre les appareils physiques, à gauche sur la Figure 2, en fonction des instances du ConnectionPointType NetworkType et des relations décrites. ConnectionPointType et NetworkType sont spécifiés dans l'IEC 62541-100.

5 Paquetage de Communication FDI®

5.1 Généralités

Le Serveur FDI® importe le Paquetage de Communication FDI® comme tout autre Paquetage d'Appareil FDI®. L'Article 5 spécifie les informations détaillées du Paquetage de Communication FDI®.

5.2 EDD

5.2.1 Règles générales

L'EDD contenu dans le Paquetage de Communication FDI® n'est pas limité, mais lié à un profil spécifique au protocole (voir l'IEC 62769-4:2023, Annexe F).

Les éléments EDD spécifiés dans les documents de profils spécifiques à un protocole (voir l'IEC 62769-4:2023, Annexe F) et fournis avec un Paquetage de Communication FDI® doivent décrire:

a) Le paramètre et les structures de paramètre. Les définitions des paramètres obligatoires spécifiques au protocole se trouvent dans les documents de profil spécifiques au protocole (voir l'IEC 62769-4:2023, Annexe F). Les paramètres doivent contenir tout paramètre nécessitant un ajustement pour assurer le bon fonctionnement du service de communication.

b) L'identification de la Couche Physique. Des définitions spécifiques au protocole se trouvent dans l'IEC 62769-4:2023, Annexe F.

c) La modularité des Appareils de Communication: Les informations de modularité doivent être fondées sur les constructions COMPONENT de l'EDDL (voir l'IEC 61804-3).

   La FDI® représente la modularité d'un Appareil de Communication susceptible de s'adapter au matériel de communication qui fournit plusieurs voies de communication physiques ou logiques pour accéder aux multiples réseaux de communication logiques ou physiques. Chaque élément de module de l'ensemble de l'Appareil de Communication doit être décrit par un élément EDD distinct.

d) La définition du COMPONENT doit être utilisée pour prendre en charge la configuration de topologie mise en œuvre dans le système. Des définitions spécifiques au protocole se trouvent dans les documents de profil spécifiques au protocole (voir l'IEC 62769-4:2023, Annexe F). Les définitions du COMPONENT associées sont décrites en 5.2.2, 5.2.3, 5.2.4 et 5.2.7.

e) La Logique Applicative doit contenir une méthode qui permet de valider le réseau (voir 5.2.8). La fonction de validation ne prend en considération que les éléments directement connectés au réseau. La fonction de validation doit être référencée par l'attribut CHECK_CONFIGURATION spécifié par l'EDDL.

f) La Logique Applicative peut contenir une méthode qui permet de valider la configuration du module (voir 5.2.9) ou la configuration du réseau (voir 5.2.8). La fonction de validation ne prend en considération que les éléments directement connectés à l'élément parent concerné dans la topologie. La fonction de validation doit être référencée par l'attribut CHECK_CONFIGURATION spécifié par l'EDDL.

g) Données relatives au Point de Connexion: Le Point de Connexion (voir 5.2.4 et 5.2.6) doit être décrit par les constructions EDDL COMPONENT, COLLECTION et VARIABLE. La définition du COMPONENT associe l'élément Point de Connexion à l'Appareil de Communication. Les définitions de la VARIABLE représentent les propriétés d'un Point de Connexion spécifique. La COLLECTION représente la structure du Point de Connexion comme telle. Des définitions spécifiques au protocole se trouvent dans l'IEC 62769-4:2023, Annexe F. L'Annexe A décrit une convention applicable à la création d'annexes spécifiques au protocole.

h) MENU:

   La structure du Menu doit respecter les conventions relatives au Menu pour les applications sur PC conformément à l'IEC 61804-4, permettant ainsi l'accès aux

   1) paramètres de Type d'Appareil de Communication FDI® (bus): Ces paramètres doivent être rendus accessibles par "offline_root_menu";

   2) Les Dialogues de Configuration de la Topologie doivent être rendus accessibles par le biais du point d'entrée "topology_configuration" du menu.

5.2.2 Composant Appareil

Chaque Paquetage de Communication FDI® doit contenir un élément EDD qui décrit l'appareil.

```
COMPONENT <DeviceComponentId>
{
    LABEL "<Label>";
    CAN_DELETE TRUE;
    CHECK_CONFIGURATION <ValidateModules>;
    CLASSIFICATION NETWORK_COMPONENT;
    COMPONENT_RELATIONS
    {
        <CommunicationDeviceRelationId>
    }
}



COMPONENT_RELATION <CommunicationDeviceRelationId>
{
  LABEL "Relation type description";
  RELATION_TYPE CHILD_COMPONENT;
  ADDRESSING {<AddressVar>}
  COMPONENTS
  {
      <CommunicationDeviceComponentId>
   {
        AUTO_CREATE <autoCreate>;
        REQUIRED_RANGES
```

```
            {
              <AddressVar>{ MIN_VALUE <AddrMin>; MAX_VALUE <AddrMax>;}
    }
}
  }
   MINIMUM_NUMBER <minNumber>;
   MAXIMUM_NUMBER <maxNumber>;
}
```

<DeviceComponentId>: L'identificateur du COMPONENT identifie la description du composant pour le type d'appareil.

<Label>: La chaîne de valeur doit contenir une chaîne qui permet à un utilisateur humain de déterminer la fonction de l'objet du Serveur de Communication FDI®.

<ValidateModules>: La Valeur désigne la METHOD qui met en œuvre la fonction de validation de la configuration de topologie du module. (Les détails de mise en œuvre sont spécifiés en 5.2.9.)

L'attribut COMPONENT_RELATIONS permet de décrire comment les modules peuvent être connectés. La définition de COMPONENT_RELATIONS est facultative. Si elle est utilisée, elle doit décrire les relations aux définitions de CommunicationDevice. La construction permet de procéder à une configuration générique de la topologie (de l'appareil) pilotée par le Serveur FDI®. Les détails syntaxiques sont décrits dans l'IEC 61804-3. Le texte suivant décrit l'usage sémantique de la construction COMPONENT_RELATION.

<CommunicationDeviceRelationId>: La valeur d'attribut identifie la définition COMPONENT_RELATION qui décrit la relation entre le composant Device et le composant CommunicationDevice.

<CommunicationDeviceComponentId>: Il est nécessaire que la valeur d'attribut corresponde à l'identificateur du COMPONENT utilisé dans une déclaration COMPONENT qui décrit un Appareil de Communication (voir 5.2.3).

<autoCreate>: La valeur d'attribut décrit le nombre de composants CommunicationDevice qui peuvent être automatiquement instanciés avec le composant Device.

<minNumber>/<maxNumber>/<autoCreate>: Les valeurs des attributs définissent les contraintes d'instanciation. La définition de ces attributs est facultative. Les valeurs des attributs peuvent contenir des expressions conditionnelles.

Le RELATION_TYPE doit être défini sur CHILD_COMPONENT.

<AddressVar>: La valeur d'attribut est une référence à une déclaration de VARIABLE. Cette VARIABLE contient la valeur d'adresse pour une instance CommunicationDevice. La définition de cet attribut est facultative.

<AddrMin>/<AddrMax>: Les valeurs définissent la plage de valeurs d'adresse pour une instance CommunicationDevice. La valeur peut, par exemple, correspondre à un numéro d'emplacement physique. L'utilisation des attributs ADDRESSING et REQUIRED_RANGES active les routines de configuration générique.

5.2.3 Composant CommunicationDevice

Chaque Paquetage de Communication FDI® doit contenir au moins un élément EDD qui décrit au moins un composant CommunicationDevice. La structure d'un matériel de communication modulaire doit être décrite par des descriptions COMPONENT CommunicationDevice multiples:

```
COMPONENT <CommunicationDeviceComponentId>
{
    LABEL "<Label>";
    CAN_DELETE <CanDelete>;
    CLASSIFICATION NETWORK_COMPONENT;
    COMPONENT_RELATIONS
    {
<CommunicationServiceProviderRelationId>
    }
}



COMPONENT_RELATION <CommunicationServiceProviderRelationId>
{
  LABEL "Relation between CommunicationDevice and communication
service provider";
  RELATION_TYPE CHILD_COMPONENT;
  ADDRESSING {<AddressVar>}
  COMPONENTS
  {
      <CommunicationServiceProviderId>
   {
        AUTO_CREATE <autoCreate>;
}
  }
   MINIMUM_NUMBER 1;
   MAXIMUM_NUMBER <maxNumber>;
}
```


<CommunicationDeviceComponentId>: L'identificateur du COMPONENT identifie le composant CommunicationDevice.

<Label>: La valeur de chaîne doit contenir une chaîne interprétable par l'homme qui permet à un utilisateur de déterminer facilement la fonction du composant CommunicationDevice.

<CanDelete>: Les valeurs autorisées sont TRUE ou FALSE. Tout dépend si un CommunicationDevice a besoin d'une configuration explicite ou si l'objet de fournisseur de service de communication associé doit être instancié automatiquement avec le CommunicationDevice. Lorsque l'attribut CAN_DELETE est défini sur FALSE, la configuration de CommunicationDevice est statique.

La définition de COMPONENT_RELATIONS est obligatoire. Elle décrit la relation à la définition du fournisseur de services de communication. La construction permet au Serveur FDI® d'instancier des composants Fournisseur de services de communication en fonction des demandes de traitement de communication (les détails syntaxiques sont décrits dans l'IEC 61804-3). Le texte suivant décrit l'usage sémantique de la construction COMPONENT_RELATION.

<CommunicationServiceProviderRelationId> La valeur d'attribut identifie la définition COMPONENT_RELATION en tant que telle.

<CommunicationServiceProviderId>: La valeur d'attribut doit correspondre à l'identificateur du COMPONENT utilisé dans une déclaration COMPONENT qui décrit un fournisseur de services de communication (5.2.4).

<autoCreate>: La valeur d'attribut décrit le nombre de fournisseurs de services de communication qui peuvent être automatiquement instanciés avec le composant CommunicationDevice.

Le RELATION_TYPE doit être défini sur CHILD_COMPONENT.

L'attribut PROTOCOL ne doit pas être défini.

5.2.4 Composant fournisseur de services de communication

Chaque Paquetage de Communication FDI® qui décrit un Appareil de Communication doit contenir au moins un élément EDD qui décrit le fournisseur de services de communication. Le composant EDD ne doit définir aucun paramètre de configuration.

```
COMPONENT <CommunicationServiceProviderId>
{
    LABEL "<Label>";
    BYTE_ORDER <byteOrder>;
    CAN_DELETE <CanDelete>;
    CLASSIFICATION NETWORK_COMMUNICATION_SERVICE_PROVIDER;
    COMPONENT_RELATIONS
    {
<CommunicationServiceProvidersConnectionPointRelationId>
    }
}

COMPONENT_RELATION
<CommunicationServiceProvidersConnectionPointRelationId>
{
  LABEL "Relation between communication service provider and
connection point";
  RELATION_TYPE CHILD_COMPONENT;
  ADDRESSING {<AddressVar>}
  COMPONENTS
  {
      < ConnectionPointId>
   {
       AUTO_CREATE 1;
}
  }
   MINIMUM_NUMBER 1;
   MAXIMUM_NUMBER 1;
}
```


<CommunicationServiceProviderId>: L'identificateur du COMPONENT identifie le fournisseur de services de communication.

<Label>: La valeur de chaîne doit contenir une chaîne interprétable par l'homme qui permet à un utilisateur de déterminer facilement la fonction de l'objet du fournisseur de services de communication.

<CanDelete>: Les valeurs autorisées sont TRUE ou FALSE. Tout dépend si un fournisseur de service de communication peut être instancié de manière flexible selon les demandes de traitement de communication. Si l'attribut CAN_DELETE est défini sur FALSE, le nombre d'instanciations du composant du fournisseur de services de communication est statique. Les contraintes d'instanciation déclarées par les attributs AUTO_CREATE, MINIMUM_NUMBER et MAXIMUM_NUMBER correspondent aux capacités des protocoles actuellement pris en charge.

<byteOrder>: La valeur permet une intégration générique des types de données à n octets (par exemple, entier de 4 octets) à la charge utile du message de communication. La valeur d'attribut décrit l'ordre des octets et doit être BIG_ENDIAN ou LITTLE_ENDIAN.

La définition de COMPONENT_RELATIONS est obligatoire. Elle décrit la relation à la définition du Point de Connexion. La construction permet de procéder à une configuration générique de la topologie pilotée par le Serveur FDI® (les détails syntaxiques sont décrits dans l'IEC 61804-3). Le texte suivant décrit l'usage sémantique de la construction COMPONENT_RELATION.

Le Point de Connexion doit automatiquement être instancié avec le fournisseur de services de communication et il doit y avoir exactement une (1) instance de Point de Connexion connectée au fournisseur de services de communication. Les contraintes d'instanciation déclarées par les attributs AUTO_CREATE, MINIMUM_NUMBER et MAXIMUM_NUMBER correspondent aux capacités des protocoles actuellement pris en charge.

<CommunicationServiceProvidersConnectionPointRelationId> La valeur d'attribut identifie la déclaration COMPONENT_RELATION en tant que telle.

<ConnectionPointId>: La valeur d'attribut doit correspondre à l'identificateur du COMPONENT utilisé dans une déclaration COMPONENT qui décrit un Point de Connexion (voir 5.2.5).

Le RELATION_TYPE doit être défini sur CHILD_COMPONENT.

L'attribut PROTOCOL ne doit pas être défini.

5.2.5 Composant Point de Connexion

Chaque Paquetage de Communication FDI® qui décrit un Appareil de Communication doit contenir un élément EDD qui décrit un Point de Connexion pour chacun des protocoles pris en charge par l'Appareil de Communication:

```
COMPONENT <ConnectionPointId>
{
    LABEL "<Label>";
    CAN_DELETE FALSE;
    CLASSIFICATION NETWORK_CONNECTION_POINT;
    PROTOCOL <ProtocolId>;
    CONNECTION_POINT <ConnectionPointCollectionId>;
}
```

<ConnectionPointId>: L'identificateur du COMPONENT identifie la déclaration de composants de Point de Connexion.

<Label>: La chaîne de valeur doit contenir une chaîne qui permet à un utilisateur humain de déterminer la fonction du composant Point de Connexion.

<ProtocolID>: La valeur de cet attribut indique la capacité de communication qui permet au Serveur FDI® de trouver d'autres types d'appareils pouvant être connectés au réseau par le même type de protocole. Concernant les protocoles normalisés, la valeur est définie par l'organisme fieldbus concerné.

<ConnectionPointCollectionId>: La valeur d'attribut est une référence à une déclaration de COLLECTION qui décrit la structure de données du Point de Connexion, comme cela est décrit en 5.2.6.

5.2.6 Collection Point de Connexion

Chaque EDD qui décrit le Point de Connexion d'un Appareil de Communication doit décrire l'élément COLLECTION qui décrit les attributs qui doivent apparaître dans la représentation du Modèle d'Information du Point de Connexion. Les données spécifiques à un protocole exposées par le Point de Connexion identifient le type d'appareil et son adresse réseau.

```
COLLECTION <ConnectionPointCollectionId>
{
    LABEL "<Label>";
    MEMBERS
    {
        <AddressAttributeName>, <AddressAttributeVariableId>;
        VALID <VALID_VariableId>;
    }
}
```

<ConnectionPointCollectionId>: L'identifiant de la COLLECTION est référencé par la valeur d'attribut CONNECTION_POINT définie en 7.7.3.5.

<Label>: L'étiquette identifie le Point de Connexion dans un langage interprétable par l'homme.

<AddressAttributeName>/<AddressAttributeVariableId>: La section MEMBRE désigne les définitions de VARIABLE qui décrivent les attributs d'adresse mis en œuvre par un Point de Connexion. Le contenu de la section MEMBRE est spécifique au protocole.

<VALID>/<VALID_VariableId> est un membre de Collection qui désigne une VARIABLE booléenne qui contient l'état de validation qui doit être défini par l'Action ValidateNetwork (voir 5.2.8).

5.2.7 Composant réseau

Chaque Paquetage de Communication FDI® qui décrit un Appareil de Communication doit contenir un élément EDD qui décrit un Réseau pour chacun des protocoles pris en charge par l'Appareil de Communication. La définition prend en charge l'ingénierie topologique du réseau.

```
COMPONENT <NetworkComponentId>
{
    LABEL "<Label>";
    CAN_DELETE TRUE;
    CHECK_CONFIGURATION <Validate>;
    CLASSIFICATION NETWORK;
    PROTOCOL <ProtocolId>;
    COMPONENT_RELATIONS
    {
        <NetworksConnectionPointRelationId>
    }
}
```

```
COMPONENT_RELATION <NetworksConnectionPointRelationId>
{
  LABEL "Relation between network and connection point";
  RELATION_TYPE CHILD_COMPONENT;
  ADDRESSING {<AddressVar>}
  COMPONENTS
  {
      <ConnectionPointId>
   {
        REQUIRED_RANGES
           {
             <BusAddressVar>{ MIN_VALUE <BusAddrMin>; MAX_VALUE
<BusAddrMax>;}
    }
}
  }
   MINIMUM_NUMBER 1;
   MAXIMUM_NUMBER <maxNumber>;
}
```

<NetworkComponentId>: L'identificateur du COMPONENT identifie la déclaration de composants de Réseau.

<Label>: La valeur de chaîne doit contenir une chaîne interprétable par l'homme qui permet à un utilisateur de déterminer facilement la fonction du composant Network.

<Validate>: La Valeur désigne la METHOD qui met en œuvre la fonction de validation de la configuration de topologie du réseau (voir 5.2.8).

<ProtocolID>: La valeur de cet attribut permet au Serveur FDI® de trouver d'autres types d'appareils pouvant être connectés au réseau par le même type de protocole. Concernant les protocoles normalisés, la valeur est définie par l'organisme fieldbus concerné.

La définition de COMPONENT_RELATIONS est obligatoire. Elle décrit la relation à la définition du Point de Connexion et, par conséquent, les capacités d'un réseau. La construction permet de procéder à une configuration générique de la topologie de réseau pilotée par le Serveur FDI®. Les détails syntaxiques sont décrits dans l'IEC 61804-3. Le texte suivant décrit l'usage sémantique de la construction COMPONENT_RELATION.

<NetworksConnectionPointRelationId> La valeur d'attribut identifie la définition COMPONENT_RELATION.

<ConnectionPointId>: La valeur d'attribut doit correspondre à l'identificateur du COMPONENT utilisé dans une déclaration COMPONENT qui décrit un Point de Connexion (voir 6.2.4).

<maxNumber: La valeur d'attribut limite le nombre de Points de Connexion qui peuvent être connectés au réseau. Les valeurs des attributs peuvent contenir des expressions conditionnelles.

Le RELATION_TYPE doit être défini sur CHILD_COMPONENT.

<BusAddressVar>: La valeur d'attribut est une référence à une déclaration de VARIABLE. Cette VARIABLE contient la valeur de l'adresse réseau pour tout appareil connecté au réseau.

<BusAddrMin>/<BusAddrMax>: Les valeurs définissent la plage de valeurs d'adresse réseau.

5.2.8 ValidateNetwork

La méthode ValidateNetwork représente la Logique Applicative mise en œuvre dans l'Appareil de Communication qui valide une topologie réseau réelle. La méthode ValidateNetwork gère toutes les dépendances nécessaires relatives aux paramètres du bus. La mise en œuvre de la logique EDDL concernée repose sur la fonction Builtin ObjectReference de l'EDDL, qui permet d'analyser un ensemble d'instances enfant (instances du Point de Connexion). La logique de validation doit définir l'attribut <VALID> de l'instance du Point de Connexion qui a satisfait à la validation.

La mise en œuvre de ValidateModules est facultative si la configuration de module est statique ou si les règles de configuration définies dans la construction COMPONENT suffisent à configurer le module.

Le Tableau 1 fournit les arguments de l'Action ValidateNetwork.

Signature

```
ValidateNetwork(
  [out] Int32  ServiceError,
  [out] String ErrorMessage);
```

Tableau 1 – Arguments de l'Action ValidateNetwork

Argument	Description
ServiceError	0: OK -1: Echec/le Point de Connexion qui n'a pas passé la validation est indiqué par la valeur d'attribut () <VALID> définie sur false. Remarque: Les valeurs d'argument correspondent aux codes d'erreur spécifiés dans l'IEC 61804-3 qui sont dénommés BI_SUCCESS (valeur = 0) et BI_ERROR (valeur = -1). L'Action retourne le résultat ServiceError au moyen de l'énoncé "return".
ErrorMessage	Si la méthode renvoie une chaîne vide (NULL), l'appel d'Action a réussi. En cas d'erreur, l'Action peut retourner une description du problème.

5.2.9 ValidateModules

La méthode ValidateModules valide l'installation actuelle du module. La mise en œuvre de la logique EDDL concernée repose sur la fonction Builtin ObjectReference de l'EDDL, qui permet d'analyser un ensemble d'instances enfant. La mise en œuvre de ValidateModules est facultative si la configuration de module est statique ou si les règles de configuration définies dans la construction COMPONENT suffisent à configurer le module.

NOTE La décision selon laquelle ValidateModules est nécessaire ou non dépend du fournisseur.

Le Tableau 2 fournit les arguments de l'Action ValidateModules.

Signature

```
ValidateModules(
  [out] Int32  serviceError,
  [out] String ErrorMessage);
```

Tableau 2 – Arguments de l'Action ValidateModules

Argument	Description
ServiceError	0: OK -1: Echec/le Point de connexion qui n'a pas passé la validation est indiqué par la valeur d'attribut () <VALID> définie sur false. Remarque: Les valeurs d'argument correspondent aux codes d'erreur spécifiés dans l'IEC 61804-3, qui sont dénommés BI_SUCCESS (valeur = 0) et BI_ERROR (valeur = -1). L'Action retourne le résultat ServiceError au moyen de l'énoncé "return".
ErrorMessage	Si l'Action renvoie une chaîne vide (NULL), l'appel de méthode a réussi. En cas d'erreur, l'Action peut retourner une description du problème.

5.2.10 Eléments spécifiques de l'UIP

Le Paquetage de Communication FDI® peut contenir l'UIP afin de prendre en charge, par exemple, les diagnostics et le paramétrage.

5.2.11 Déploiement

Le Serveur FDI® importe le Paquetage de Communication FDI®. Le traitement des pièces EDD et UIP obéit à la procédure d'importation appliquée pour le Paquetage FDI® (voir l'IEC 62769-2 et l'IEC 62769-3).

6 Relations de communication

L'objectif d'un Appareil de Communication et de ses services de communication est d'échanger les informations entre l'appareil physique et la représentation de l'appareil gérée par le Serveur FDI®. L'échange d'informations est géré par les relations de communication (voir Figure 3). Une relation de communication établie permet l'échange d'informations entre la représentation de l'appareil gérée par le Serveur FDI® et l'appareil physique. L'utilisation de la Relation de communication permet de faire abstraction des éléments spécifiques du protocole généralement utilisés pour gérer les connexions.

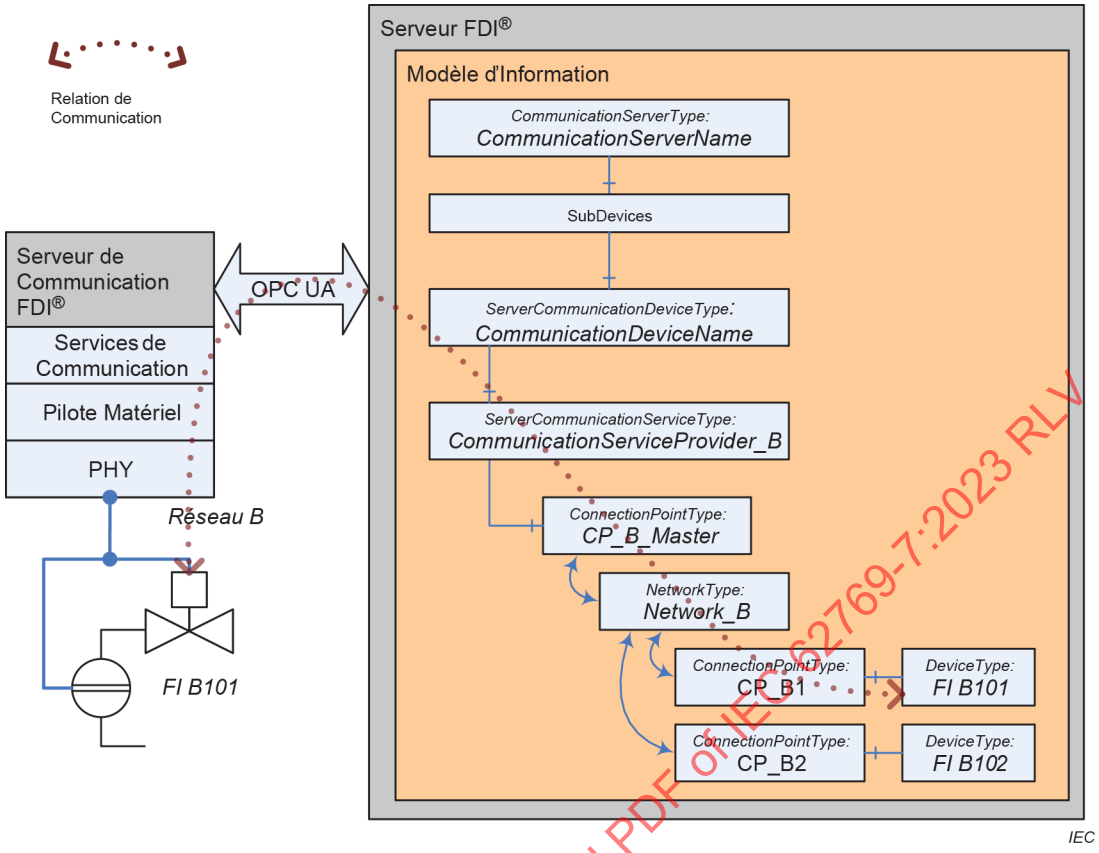


Figure 3 – Relation de communication

NOTE 1 L'essentiel de l'échange d'informations a lieu entre l'appareil connecté au réseau physique et l'instance correspondante dans le Modèle d'Information, mais ne couvre pas l'application complète de l'appareil.

Le diagramme d'états suivant décrit le flux d'états général d'une relation de communication. Le diagramme indique également les services de communication qui peuvent être appelés pendant un état "CR Online".

"AbortIndication" indiqué à la Figure 4 peut être détecté de différentes manières selon le protocole. La façon spécifiée pour un Appareil de Communication donné est liée aux serviceError retournées par les services de communication spécifiés. Même la méthode de balayage (Scan) peut déterminer une perte de connexion quand l'appareil pour lequel une relation de communication a été activée n'apparaît pas dans le résultat du balayage.

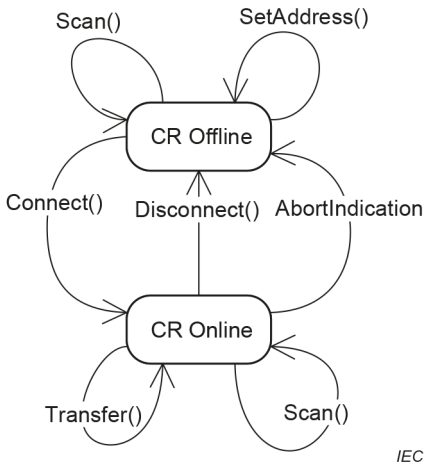


Figure 4 – Diagramme d'états-transitions de la relation de communication

NOTE 2 La gestion des relations de communication est facultative.

7 Définition du Serveur de Communication FDI®

7.1 Généralités

En matière de FDI®, le Serveur de Communication FDI® est un Serveur dédié de l'architecture unifiée OPC (OPC UA) qui fournit l'accès à un ou plusieurs réseaux d'appareils de terrain. Chaque Serveur de Communication FDI® est conçu comme un Appareil Modulaire dans lequel chaque module (appelé également CommunicationDevice dans la séquence) représente le point d'accès à un réseau.

L'Appareil Modulaire lui-même représente le Serveur de Communication FDI® comme un ensemble.

7.2 Caractéristiques générales

Le Serveur de Communication FDI® met en œuvre les caractéristiques de chacun de ses CommunicationDevice spécifiés en 7.3.3. En outre, le Serveur de Communication FDI® met en œuvre les caractéristiques suivantes:

- le Serveur FDI® synchronise en permanence (voir 7.5, 7.8.8 et 7.8.11) le Modèle d'Information hébergé par le Serveur de Communication FDI® depuis le contenu du Modèle d'Information hébergé par le Serveur FDI®;
- les CommunicationDevice peuvent être instanciés statiquement ou peuvent être créés/supprimés par le Serveur FDI®;
- la communication entre le Serveur FDI® et le Serveur de Communication FDI® est fondée sur OPC UA. OPC UA spécifie un protocole filaire pour ses services qui peuvent être mis en œuvre sur les plates-formes arbitraires et les environnements d'exécution;
- pour éviter les accrochages, le Serveur de Communication FDI® permet la connexion d'un seul Serveur FDI® à la fois. Avec cette restriction, un Serveur de Communication FDI® peut s'abstenir de tout mécanisme de synchronisation (verrouillage). La spécification FDI® n'impose pas aux Serveurs de Communication FDI® de mettre en œuvre un mécanisme de verrouillage pour gérer les accès simultanés à un seul appareil connecté au réseau physique.

7.3 Modèle d'Information

7.3.1 Généralités

Le Paragraphe 7.3 spécifie le Modèle d'Information hébergé par le Serveur de Communication FDI®.

Un Serveur de Communication FDI® est un Serveur OPC UA qui encapsule le matériel de communication et fournit la capacité normalisée de communication. Le Serveur FDI® se connecte au Serveur de Communication FDI® comme Client OPC UA et accède aux réseaux pris en charge par le Serveur de Communication FDI® par le biais du Modèle d'Information du Serveur de Communication FDI®. La tâche du Serveur de Communication FDI® consiste à exposer ce Modèle d'Information. Le Serveur de Communication FDI® ne doit pas conserver les Instances d'Appareil ni les informations de topologie du réseau. Toute interaction avec les appareils FDI® a lieu par le biais du Serveur FDI® et est juste transférée par le Serveur de Communication FDI®.

Pour le Serveur FDI®, un Serveur de Communication FDI® prend la forme d'un appareil qui prend en charge les Services de Communication FDI® et utilise l'architecture unifiée OPC pour communiquer. Le Serveur de Communication FDI® peut s'exécuter en local sur le même ordinateur que le Serveur FDI® (adaptateur de bouclage) ou à distance sur le terrain (par exemple, intégré à un contrôleur). A l'instar d'un appareil, chaque Serveur de Communication FDI® est associé à un Paquetage FDI®. Ce Paquetage FDI® est utilisé pour

créer des Appareils de Communication dans le Modèle d'Information du Serveur FDI® qui représentent l'accès aux réseaux mis en œuvre par le Serveur de Communication FDI®.

Le Modèle d'Information d'un Serveur de Communication FDI® est fondé sur le Modèle d'Information défini dans l'IEC 62769-5. La Figure 5 reproduit la structure de l'Appareil modulaire et montre comment elle est mappée à l'AddressSpace complet. Les modules représentent les voies de communication du Serveur de Communication FDI®. La Figure 5 définit les BrowseNames des nœuds.

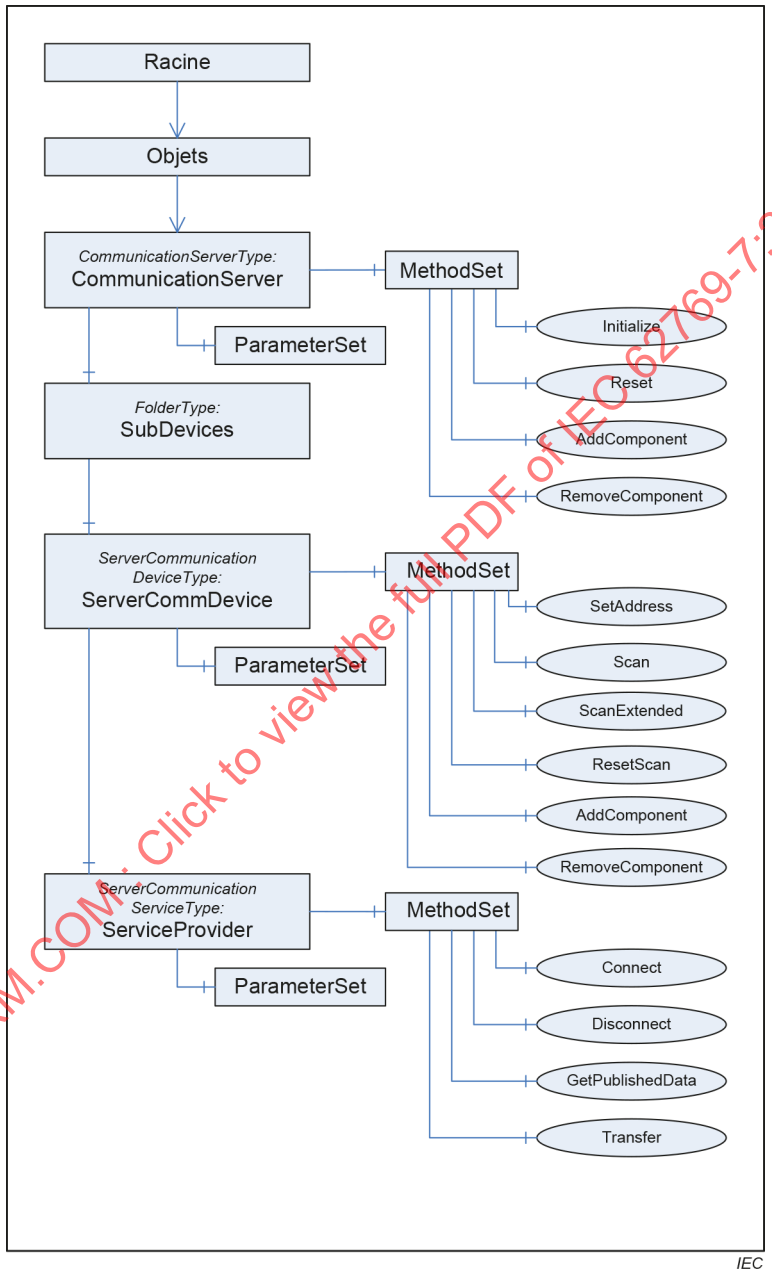


Figure 5 – AddressSpace du Serveur de Communication FDI®

Le CommunicationServerType (la racine de l'Appareil Modulaire) est un sous-type de DeviceType. Le MethodSet contient les méthodes Initialize, Reset, AddComponent et RemoveComponent. Les méthodes AddComponent et RemoveComponent sont facultativement présentes lorsque le Serveur de Communication FDI® prend en charge l'instanciation dynamique des éléments dans le dossier SubDevices.

Tous les sous-appareils sont des instances du ServerCommunicationDeviceType défini en 7.3.3. Les instances de ServerCommunicationDeviceType (ServerCommDevice) ont un MethodSet qui peut mettre en œuvre les méthodes SetAddress, Scan, AddComponent, RemoveComponent. AddComponent et RemoveComponent sont facultativement présents lorsque le Serveur de Communication FDI® prend en charge un nombre variable des instances de ServerCommunicationServiceType.

Les définitions formelles se trouvent en 7.3.2, 7.3.3 et 7.3.4.

7.3.2 CommunicationServerType

7.3.2.1 Généralités

Le CommunicationServerType est un sous-type du DeviceType et fournit les méthodes nécessaires pour gérer les instances ServerCommunicationDeviceType. La Figure 6 représente la définition de CommunicationServerType qui est formellement décrite dans le Tableau 3 et le Tableau 4.

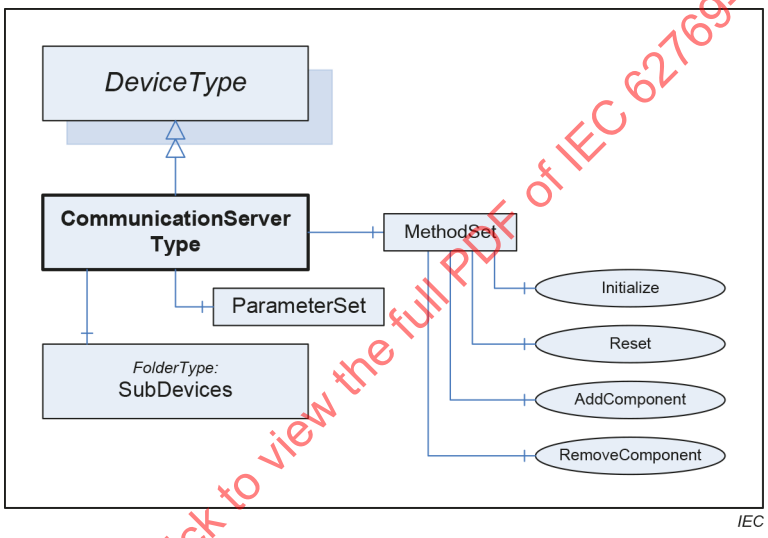


Figure 6 – CommunicationServerType

Tableau 3 – Définition de CommunicationServerType

Attribut	Valeur				
BrowseName	CommunicationServerType				
IsAbstract	False				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
Sous-type du DeviceType défini dans l'IEC 62541-100.					
HasComponent	Objet	MethodSet		BaseObjectType	Obligatoire
HasComponent	Objet	ParameterSet		BaseObjectType	Facultatif
HasComponent	Objet	SubDevices		FolderType	Obligatoire

Tableau 4 – MethodSet de CommunicationServerType

Attribut	Valeur				
BrowseName	MethodSet				
IsAbstract	False				
Références	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
HasComponent	Méthode	Initialize			Obligatoire
HasComponent	Méthode	Reset			Obligatoire
HasComponent	Méthode	AddComponent			Facultatif
HasComponent	Méthode	RemoveComponent			Facultatif

Le CommunicationServerType et chaque instance de ce Type partagent les mêmes Méthodes. Le NodeId de ces Méthodes sera fixé et défini dans le présent document (voir Annexe B). Les clients du Serveur de Communication FDI® n'ont donc pas à rechercher ces Méthodes. Ils peuvent utiliser les NodeId fixés comme MethodId du Service d'appel.

Les Méthodes supplémentaires AddComponent et RemoveComponent ajoutent ou suppriment des instances de ServerCommunicationDeviceType selon la structure du matériel de communication. Ces services ne s'appliquent pas lorsque le Serveur de Communication FDI® met en œuvre une structure de matériel de communication statique.

Le dossier SubDevices contient les instances de ServerCommunicationDeviceType qui représentent les modules de communication.

NOTE L'indication pour une disposition de matériel de communication statique est indiquée dans le Paquetage FDI® avec l'attribut COMPONENT CAN_DELETE défini sur FALSE dans les déclarations du COMPONENT.

7.3.2.2 Méthode Reset

La Méthode Reset est utilisée pour réinitialiser le matériel de communication et le logiciel du pilote concerné. Toute communication courante est arrêtée immédiatement. Tous les canaux de communication passent au statut "fermé".

La Méthode Reset ne doit pas être présente dans le Modèle d'Information hébergé par le Serveur FDI®. Le Serveur FDI® doit être capable de gérer la procédure d'arrêt automatiquement selon les demandes de communication.

Généralement, le fonctionnement du Serveur de Communication FDI® inclut le traitement du matériel et du pilote du protocole qui peut être indépendant de toute structure modulaire. En raison de cette possibilité, la Méthode Reset est mise en place sous le CommunicationServerType. Afin de réduire la complexité de fonctionnement du Serveur de Communication FDI®, une seule Méthode Reset a été spécifiée.

La signature de cette Méthode est spécifiée ci-dessous. Le Tableau 5 et le Tableau 6 spécifient les arguments et la représentation AddressSpace, respectivement.

Signature

```
Reset(
  [out] Int32  serviceError);
```

Tableau 5 – Arguments de la Méthode Reset

Argument	Description
serviceError	0: OK -1: Echec

Tableau 6 – Définition de l'AddressSpace de la Méthode Reset

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

7.3.2.3 Méthode Initialize

La Méthode Initialize sert à initialiser le matériel de communication. La fonction d'initialisation du Serveur de Communication FDI® doit utiliser les données de paramétrage hébergées par le ParameterSet qui est contenu dans l'instance du CommunicationServerType et toutes les instances de ServerCommunicationDeviceType.

Afin de permettre les modifications de paramètres pendant le fonctionnement, la méthode Initialize peut être appelée une nouvelle fois. Lorsque le Serveur de Communication FDI® nécessite de réinitialiser son matériel de communication, il doit automatiquement restaurer la relation de communication qui existait auparavant. Un Serveur de Communication FDI® modulaire ne peut initialiser de manière souple que les instances de ServerCommunicationDeviceType pour lesquelles les modifications de configuration ont été détectées.

La Méthode Initialize ne doit pas être présente dans le Modèle d'Information hébergé par le Serveur FDI®. Le Serveur FDI® doit être capable de gérer la procédure de démarrage automatiquement selon les demandes de communication humaines.

Le fonctionnement du Serveur de Communication FDI® peut inclure le traitement du matériel et du pilote du protocole qui peut être indépendant de toute structure modulaire. En raison de cette possibilité, la méthode Initialize est mise en place sous le CommunicationServerType. Afin de réduire la complexité de fonctionnement du Serveur de Communication FDI®, une seule méthode Initialize a été spécifiée.

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