BS EN 62769-7:2015

BSI Standards Publication

Field Device Integration (FDI)

Part 7: FDI Communication Devices

bsi.

...making excellence a habit.™

National foreword

This British Standard is the UK implementation of EN 62769-7:2015. It is identical to IEC 62769-7:2015.

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

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

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

Published by BSI Standards Limited 2015

ISBN 978 0 580 78329 6
ICS 25.040.40; 35.100

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

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

Amendments/corrigenda issued since publication

Date	Text affected

EUROPEAN STANDARD

NORME EUROPÉENNE

EUROPÄISCHE NORM

EN 62769-7

June 2015

ICS 25.040.40; 35.100

English Version

Field Device Integration (FDI) - Part 7: FDI Communication Devices (IEC 62769-7:2015)

Intégration des appareils de terrain (FDI) - Partie 7: Appareils de communication FDI
(IEC 62769-7:2015)

Feldgeräteintegration (FDI) - Teil 7: Kommunikationsgeräte
(IEC 62769-7:2015)

This European Standard was approved by CENELEC on 2015-06-16. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the CEN-CENELEC Management Centre or to any CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Bulgaria, Croatia, Cyprus, the Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.

CENELEC

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

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

Ref. No. EN 62769-7:2015 E

European foreword

The text of document 65E/350/CDV, future edition 1 of IEC 62769-7, prepared by SC 65E “Devices and integration in enterprise systems” of IEC/TC 65 “Industrial-process measurement, control and automation" was submitted to the IEC-CENELEC parallel vote and approved by CENELEC as EN 62769-7:2015.

The following dates are fixed:

• latest date by which the document has to be implemented at national level by publication of an identical national standard or by endorsement	(dop)	2016-03-16
• latest date by which the national standards conflicting with the document have to be withdrawn	(dow)	2018-06-16

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

Endorsement notice

The text of the International Standard IEC 62769-7:2015 was approved by CENELEC as a European Standard without any modification.

Annex ZA
(normative)

Normative references to international publications with their corresponding European publications

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

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

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

Publication	Year	Title	EN/HD	Year
IEC 61804-3	-	Function Blocks (FB) for process control -- Part 3: Electronic Device Description Language (EDDL)	EN 61804-3	-
IEC 61804-4	-	Function blocks (FB) for process control -- Part 4: EDD interpretation	-	-
IEC 62541-4	-	OPC Unified Architecture - Part 4: Services	EN 62541-4	-
IEC 62541-6	-	OPC unified architecture - Part 6: Mappings	EN 62541-6	-
IEC 62541-7	-	OPC unified architecture - Part 7: Profiles	EN 62541-7	-
IEC 62541	series	OPC Unified Architecture	EN 62541	series
IEC 62541-100	-	OPC unified architecture - Part 100: Device Interface	EN 62541-100	-
IEC 62769-1	-	Devices and integration in enterprise systems; Field Device Integration - Part 1: Overview	-	-
IEC 62769-2	-	Devices and integration in enterprise systems; Field Device Integration - Part 2: FDI Client	-	-
IEC 62769-3	-	Devices and integration in enterprise systems; Field Device Integration - Part 3: FDI Server	-	-
IEC 62769-4	2015	Devices and integration in enterprise systems; Field Device Integration - Part 4: FDI Packages	-	-
IEC 62769-5	-	Devices and integration in enterprise systems; Field Device Integration - Part 5: FDI Information Model	-	-
IEC/TR 62541-1	-	OPC unified architecture - Part 1: Overview and concepts	CLC/TR 62541-1	-

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

____

FIELD DEVICE INTEGRATION (FDI) –

Part 7: FDI 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.

International Standard 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.

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

CDV	Report on voting
65E/350/CDV	65E/420/RVD

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

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

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

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

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

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

INTRODUCTION

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

a) Method for the Supplying and Installation of Device-Specific Functionalities, see Patent Family DE10357276;

b) Method and device for accessing a functional module of automation system, see Patent Family EP2182418;

c) Methods and apparatus to reduce memory requirements for process control system software applications, see Patent Family US2013232186;

d) Extensible Device Object Model, see Patent Family US12/893,680.

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

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

a) ABB Research Ltd
Claes Rytoft
Affolterstrasse 4
Zurich, 8050
Switzerland

b) Phoenix Contact GmbH & Co KG
Intellectual Property, Licenses & Standards
Flachsmarktstrasse 8, 32825 Blomberg
Germany

c) Fisher Controls International LLC
John Dilger, Emerson Process Management LLLP
301 S. 1st Avenue, Marshaltown, Iowa 50158
USA

d) Rockwell Automation Technologies, Inc.
1 Allen-Bradley Drive
Mayfield Heights, Ohio 44124
USA

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

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

FIELD DEVICE INTEGRATION (FDI) –

Part 7: FDI Communication Devices

1 Scope

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

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

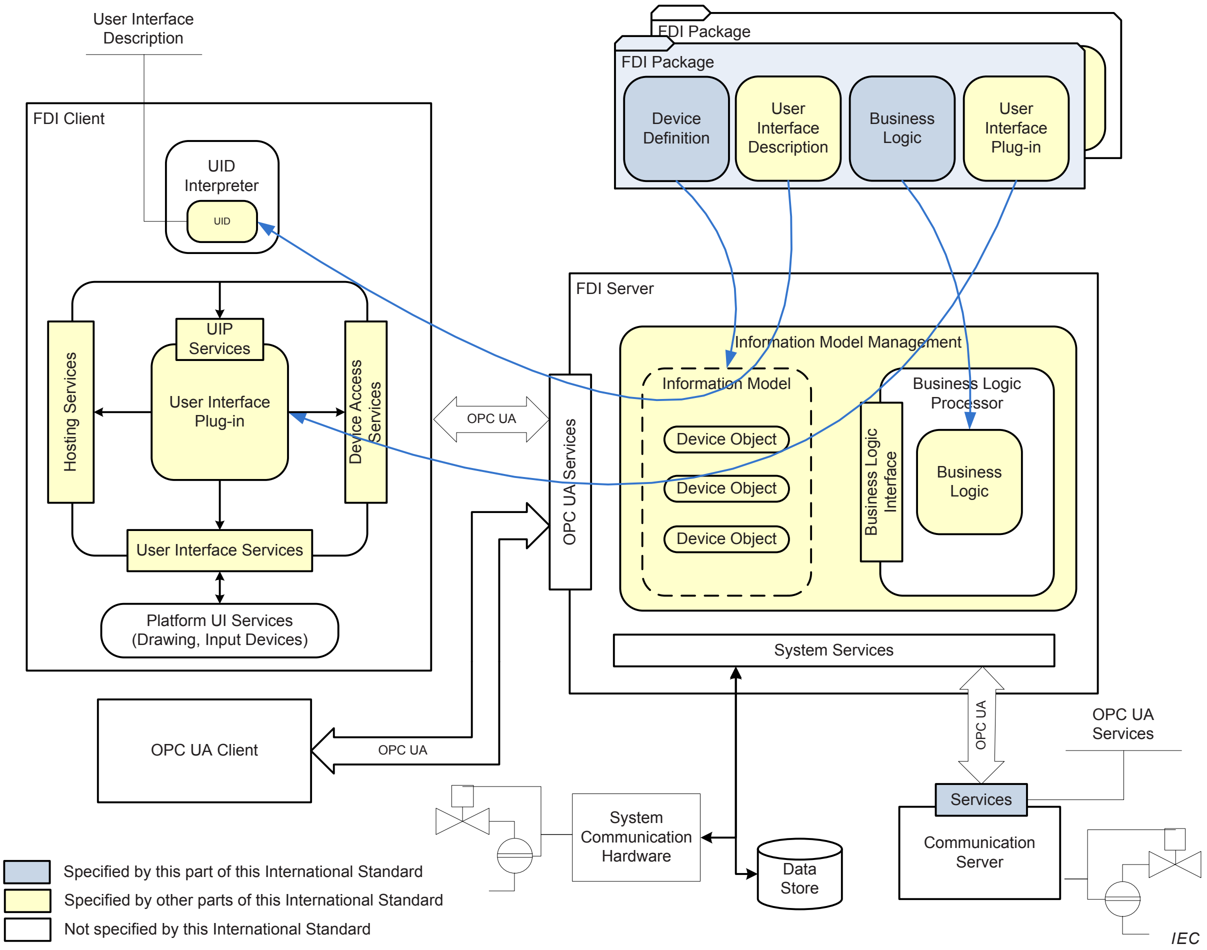


Figure 1 – FDI architecture diagram

2 Normative references

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

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

IEC 61804-4, *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: OPC UA for Devices*

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

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

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

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

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

NOTE IEC 62769-3 is technically identical to FDI-2023.

IEC 62769-4:2015, *Field Device Integration (FDI) – Part 4: FDI Packages*

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

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

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

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 as well as the following apply.

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 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 graphical notation

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

4 General

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 EED 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: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.

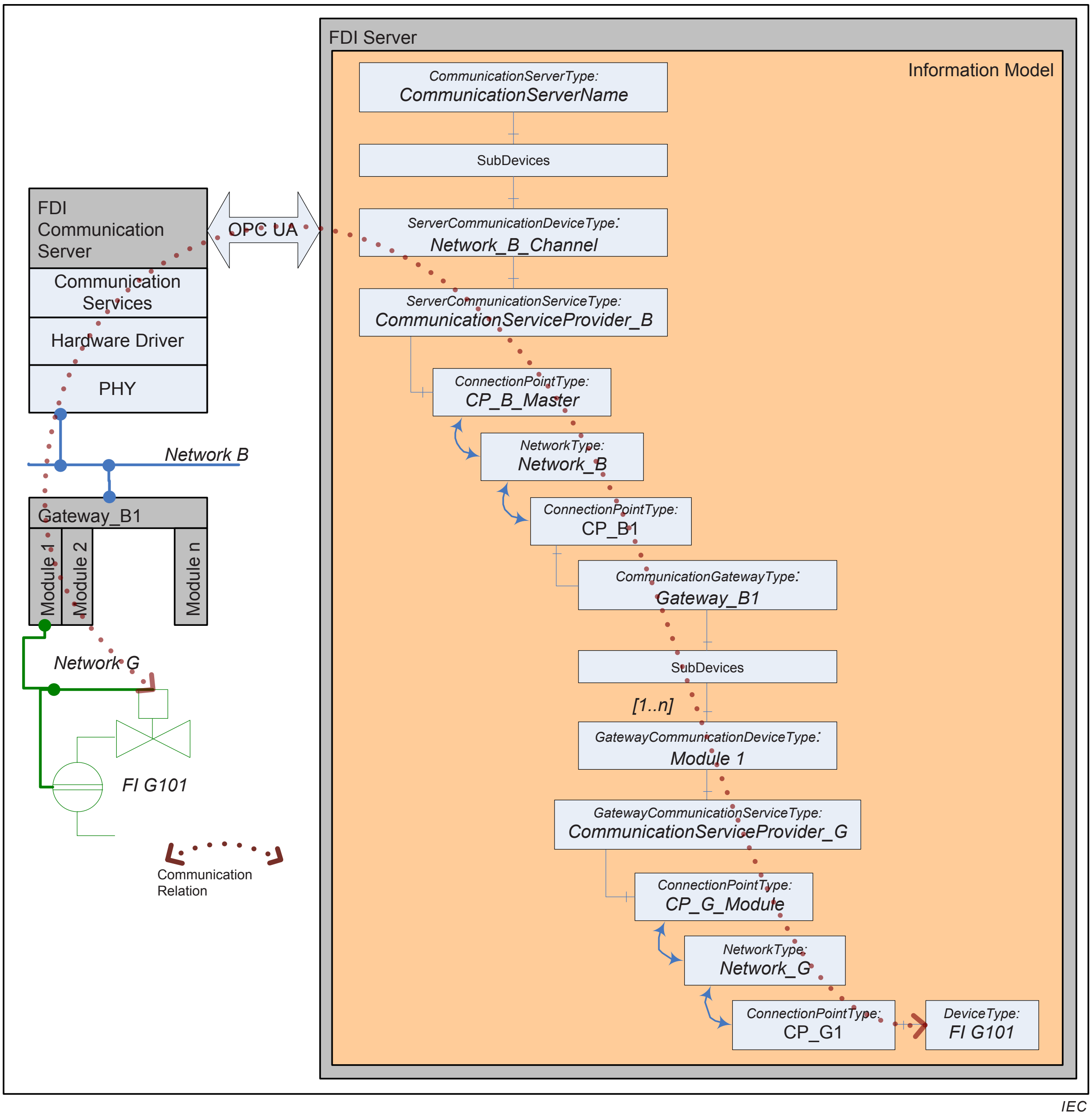


Figure 2 – FDI communication infrastructure architecture

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.

a) 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).

b) 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.

c) 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.

d) 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.

e) 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.

f) 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.

g) 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.

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 annex (IEC 62769-4:2015, Annex F).

The EDD elements as specified in IEC 62769-4:2015, Annex F, and provided with an FDI Communication Package shall describe:

a) Parameter and parameter structures. Mandatory protocol specific parameter definitions are found in IEC 62769-4:2015, Annex F. The parameter shall contain any parameter that requires adjustment for proper communication service operation.

b) Physical Layer identification. Protocol specific definitions are found in IEC 62769-4:2015, 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 IEC 62769-4:2015, 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 62769-4:2015, Annex F.

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”. Protocol specific definitions are found in IEC 62769-4:2015, Annex F.

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 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 COMPONENT_CHILD.

<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 may 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 COMPONENT_CHILD.

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>
    {
       <ConnectionPointRelationId>
    }
}

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 COMPONENT_CHILD.

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 COMPONENT_CHILD.

<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 Built-in 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 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 Built-in 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 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.

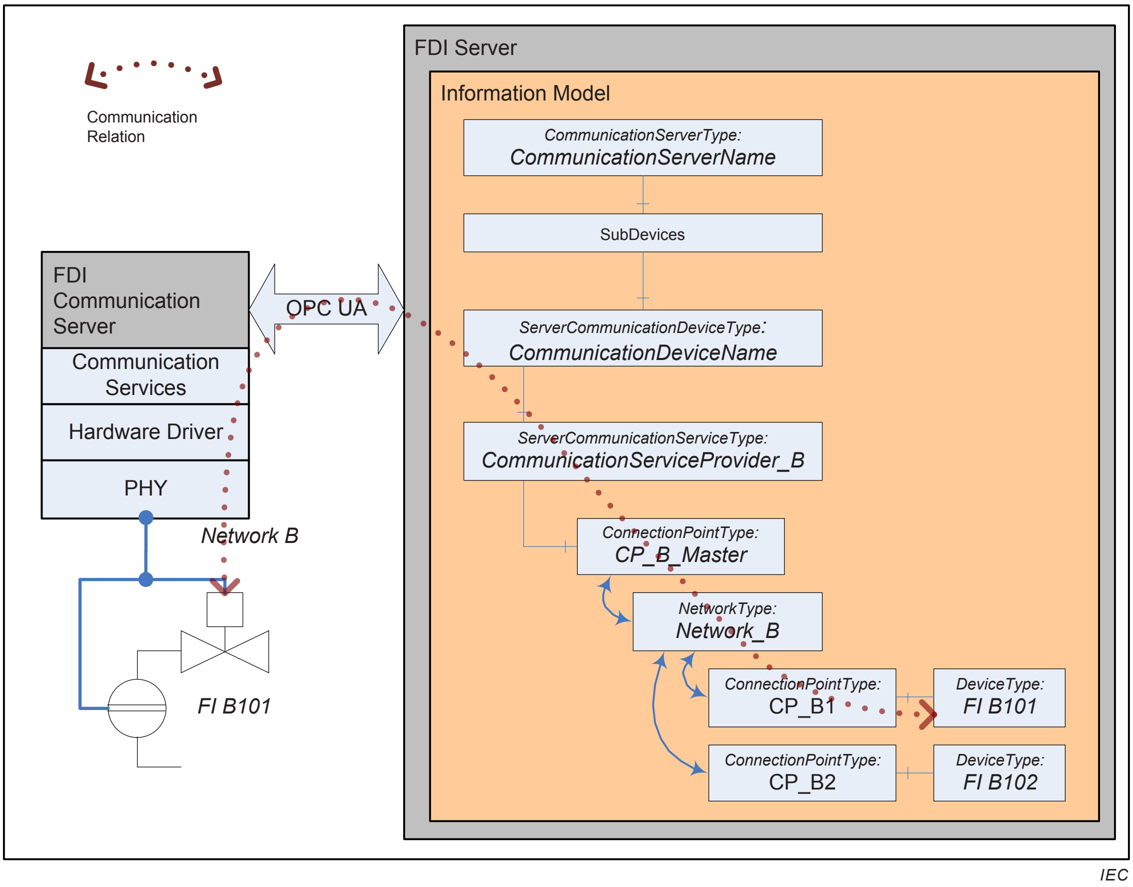


Figure 3 – Communication relation

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.

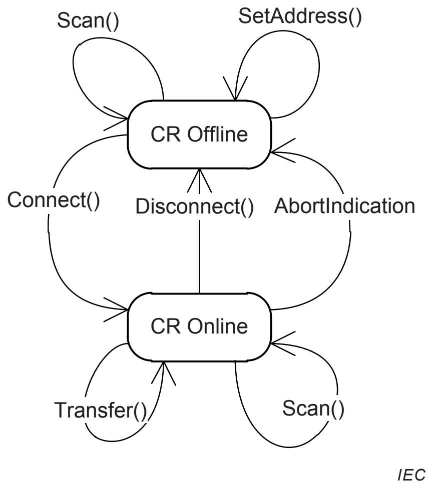


Figure 4 – Communication relation state chart

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 Sever 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 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.

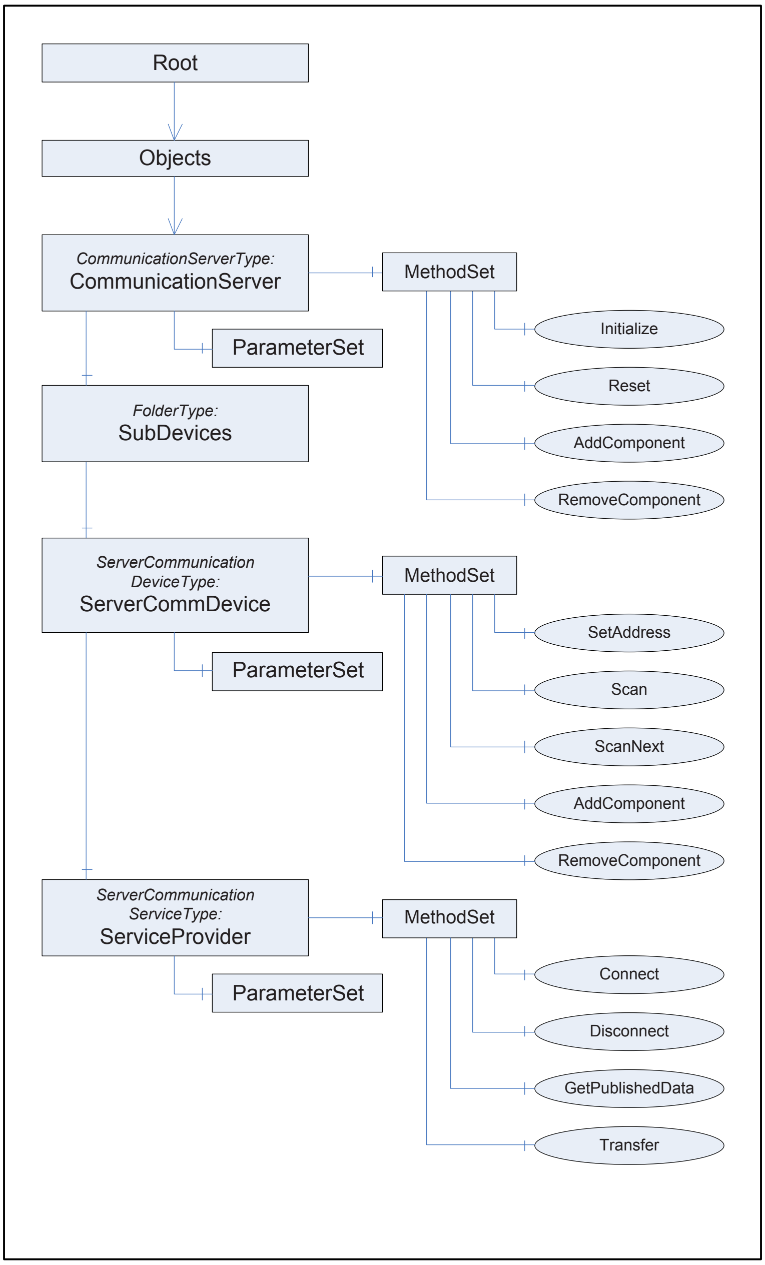


Figure 5 – FDI Communication Server AddressSpace

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.

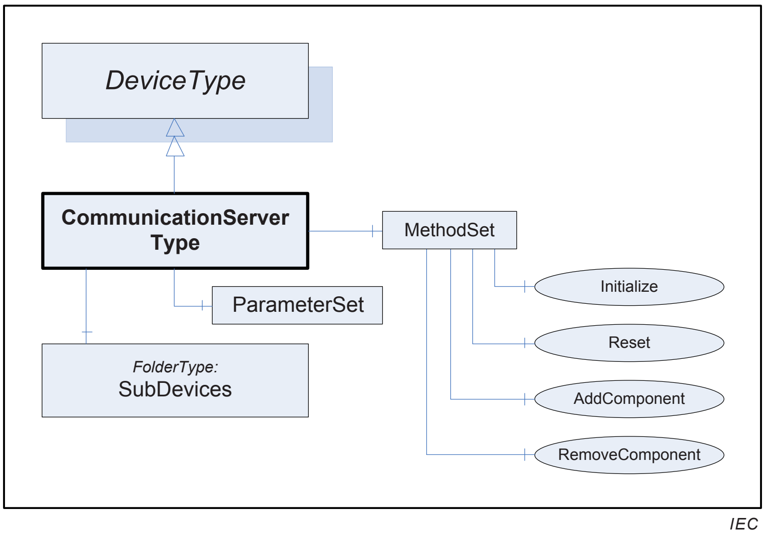


Figure 6 – CommunicationServerType

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 standard. 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        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      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      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 can not (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       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. The protocol specific, mandatory bus parameters are specified in IEC 62769-4:2015, Annex F. Figure 7 shows the ServerCommunicationDeviceType definition that is formally defined in Table 13 and Figure 14.

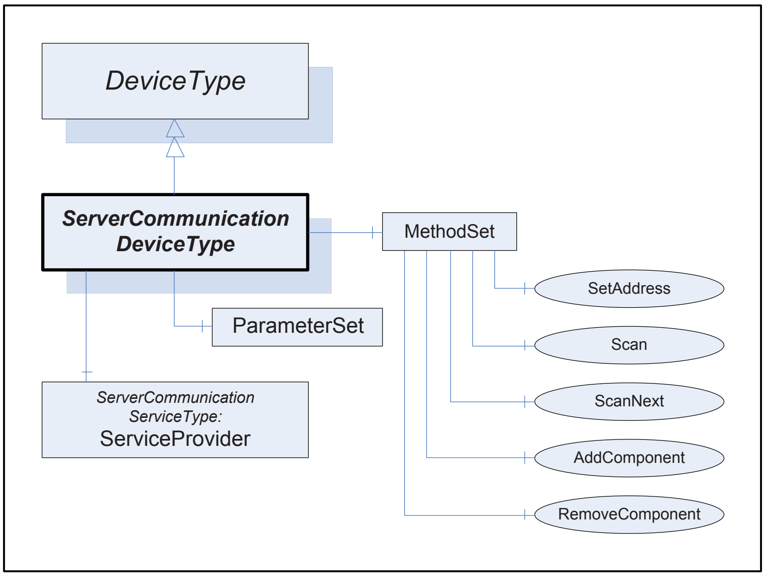


Figure 7 – ServerCommunicationDeviceType

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

Table 14– MethodSet of ServerCommunicationDeviceType

Attribute	Value				
BrowseName	MethodSet				
IsAbstract	True				
References	**NodeClass**	**BrowseName**	**DataType**	**TypeDefinition**	**ModellingRule**
HasComponent	Method	Scan			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          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 62769-4:2015, 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 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 17 and Table 18 specify the arguments and AddressSpace representation, respectively.

Signature

```
ResetScan(

  [out] Integer          ServiceError)
```

Table 17 – ResetScan Method arguments

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

Table 18 – ResetScan Method AddressSpace definition

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

7.3.3.4 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 19.

Signature

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

Table 19 – SetAddress Method arguments

Argument	Description
OldAddress	The argument represents 1..n protocol specific values representing the existing protocol specific network address of the physical network connected device. Values that represent a network address are specified in IEC 62769-4:2015, Annex F.
NewAddress	The argument represents 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 62769-4:2015, Annex F.
ServiceError	0: OK / execution finished successfully -1: SetAddress Failed / not initialized -2: SetAddress Failed / not connected to a network -3: SetAddress Failed / no device found responding to oldAddress -4: SetAddress Failed / duplicate address error -5: SetAddress Failed / device did not accept new address -6: SetAddress Failed / invalid oldAddress (in terms of syntax, data type, data format, and so on) -7: SetAddress Failed / invalid newAddress (in terms of syntax, data type, data format, and so on) -8: SetAddress Failed / not possible in status connected

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).

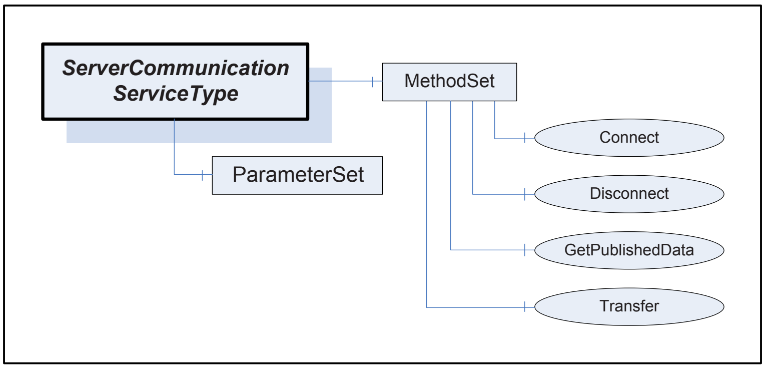
The formal definition of ServerCommunicationServiceType is found in Figure 8, Table 20 and Table 21.

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

Communication methods including their NodeIds are uniquely defined in this standard. 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.



IEC

Figure 8 – ServerCommunicationServiceType

Table 20 – ServerCommunicationServiceType definition

<table>
<tr><th>Attribute</th><th colspan="5">Value</th></tr>
<tr><td>BrowseName</td><td colspan="5">ServerCommunicationServiceType</td></tr>
<tr><td>IsAbstract</td><td colspan="5"></td></tr>
<tr><th>References</th><th>NodeClass</th><th>BrowseName</th><th>DataType</th><th>TypeDefinition</th><th>ModellingRule</th></tr>
<tr><td colspan="6">Subtype of the DeviceType defined in OPC UA Part DI.</td></tr>
<tr><td>HasComponent</td><td>Object</td><td>MethodSet</td><td></td><td>BaseObjectType</td><td>Mandatory</td></tr>
<tr><td>HasComponent</td><td>Object</td><td>ParameterSet</td><td></td><td>BaseObjectType</td><td>Optional</td></tr>
</table>

Table 21 – 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 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 62769-4:2015, 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 22 specifies the arguments.

Signature

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

Table 22 – 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	A protocol specific argument list that is used for the address and optional device identification data (details described in IEC 62769-4:2015, Annex F).
DeviceInformation	A protocol specific argument list in which the connect result data are stored.
ServiceError	0: OK / execution finished, connection established successfully -1: Connect Failed / device not found -2: Connect Failed / invalid device address -3: Connect Failed / invalid device identification

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 23. Disconnect is a synchronous method call.

Signature

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

Table 23 – 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 -1: Disconnect Failed / no existing communication relation -2: Disconnect Failed / invalid communication relation identifier

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 24.

Signature

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

Table 24 – Transfer Method arguments

Argument	Description
CommunicationRelationId	See 7.3.4.2.
SendData	A protocol specific list of values as described in IEC 62769-4:2015, Annex F. The argument values represent the protocol specific communication service request that is sent to the device.
ReceiveData	A protocol specific list of values as described in IEC 62769-4:2015, 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 -1: Transfer Failed / No existing communication relation. -2: Transfer Failed / Invalid communication relation identifier -3: Transfer Failed / Invalid sendData content -4: Transfer Failed / Invalid receiveData format

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 25 shows the GetPublishedData Method Arguments.

Signature

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

Table 25 – GetPublishedData Method arguments

Argument	Description
CommunicationRelationId	See 7.3.4.2.
SendData	A protocol specific list of values as described in IEC 62769-4:2015, Annex F. The argument values control the exchange of unsolicited messages.
ReceiveData	A protocol specific list of values as described in IEC 62769-4:2015, 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 -2: Call Failed / unknown PublishId -3: GetPublishedData Failed / not supported -4: GetPublishedData Failed / no existing communication relation. -5: GetPublishedData Failed / invalid communication relation identifier -6: GetPublishedData Failed / invalid sendData content -7: GetPublishedData Failed / invalid receiveData format -8: GetPublishedData Failed / no data published that fits to the SendData argument

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 26 describes Conformance Units included in this facet.

Table 26 – *FDICommunicationServer_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 IM 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.

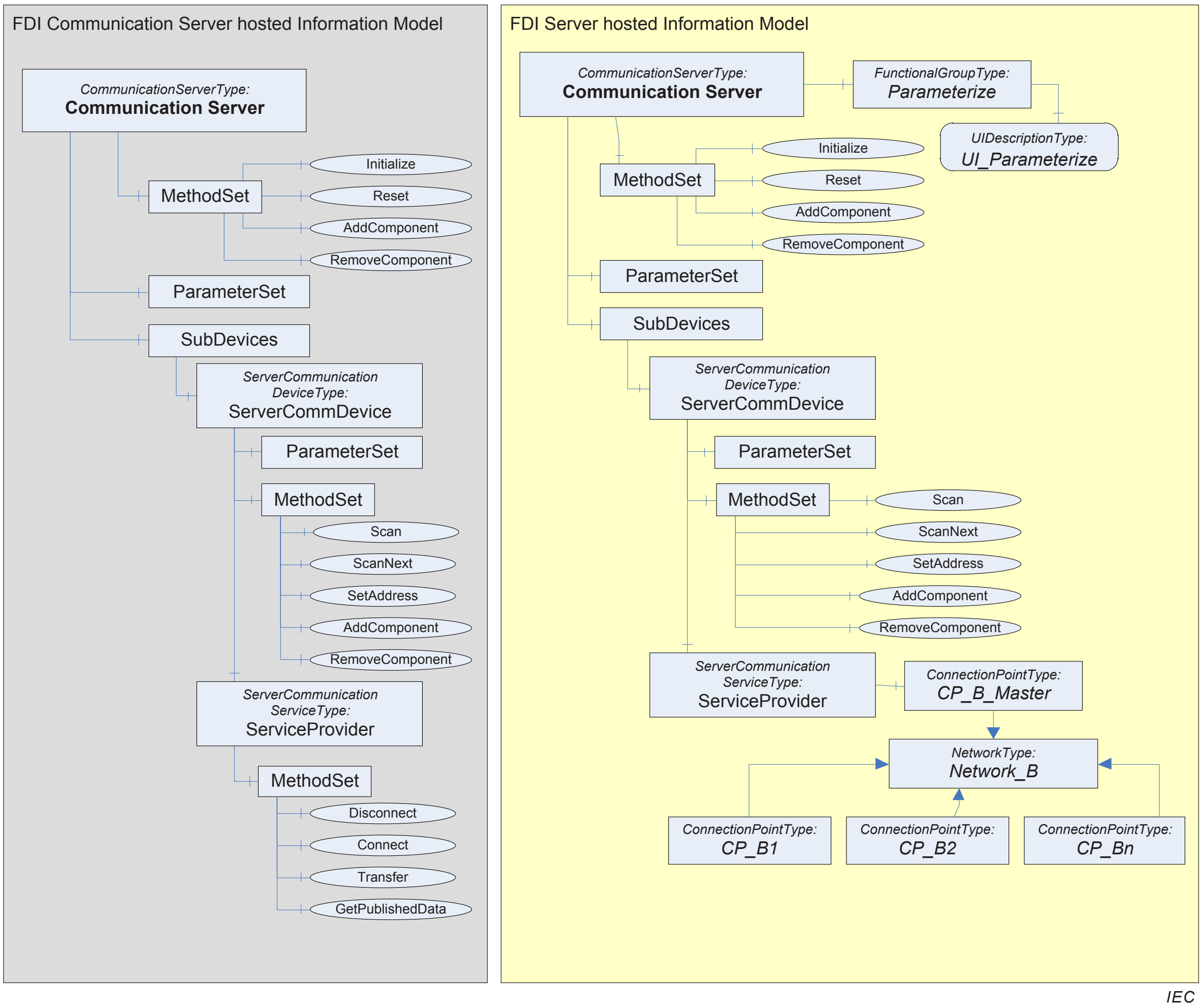


Figure 9 – Information Model differences (example)

The list of differences in the IM 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 7.7specifies the FDI Communication Package details.

With respect to the EDD element of the Package, FDI differentiates between a simple (lightweight) CommunicationServer (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 Built-in function ObjectReference

NOTE COMPONENT defines an attribute PRODUCT_URI that can be used for automatic discovery of the matching CommunicationServer. Since we are not using the COMPONENT construct for lightweight CommunicationServers, systems have to provide manual discovery.

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 define in 5.2.6 apply.

7.7.4 Documentation

The FDI Communication Package shall provide documentation describing:

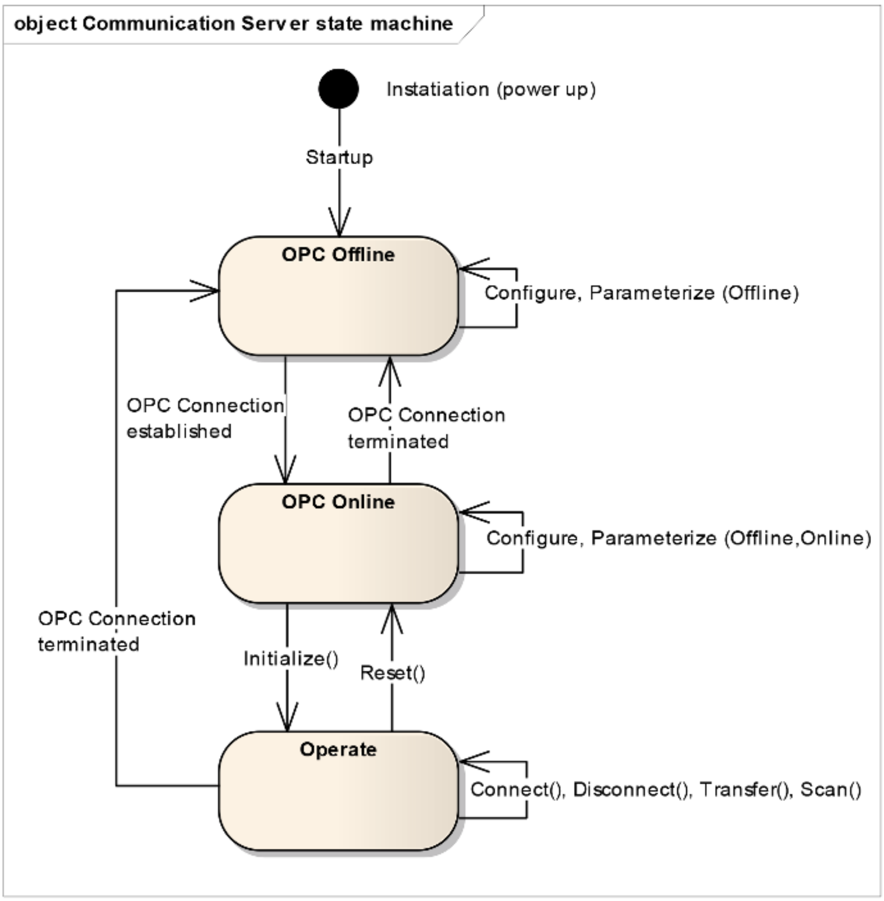
i) the software installation related environment requirements and procedures

j) the OPC UA Server configuration if needed

7.8 Handling and behavior

7.8.1 General

Subclause 7.8 defines the FDI Communication Server handling and behavior 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.


object Communication Server state machine
Instatiation (power up)
Startup
OPC Offline
Configure, Parameterize (Offline)
OPC Connection established
OPC Connection terminated
OPC Online
Configure, Parameterize (Offline,Online)
OPC Connection terminated
Initialize()
Reset()
Operate
Connect(), Disconnect(), Transfer(), Scan()

IEC

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 □) 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. The standard 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 start 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.

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 TR 62541-1. 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.

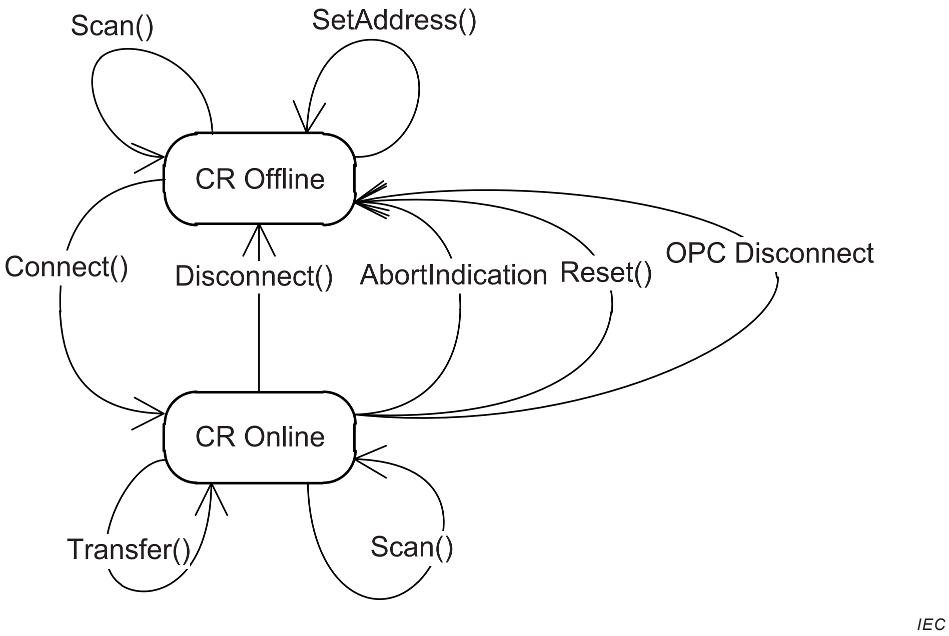


Figure 11 – Communication relation state chart

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.

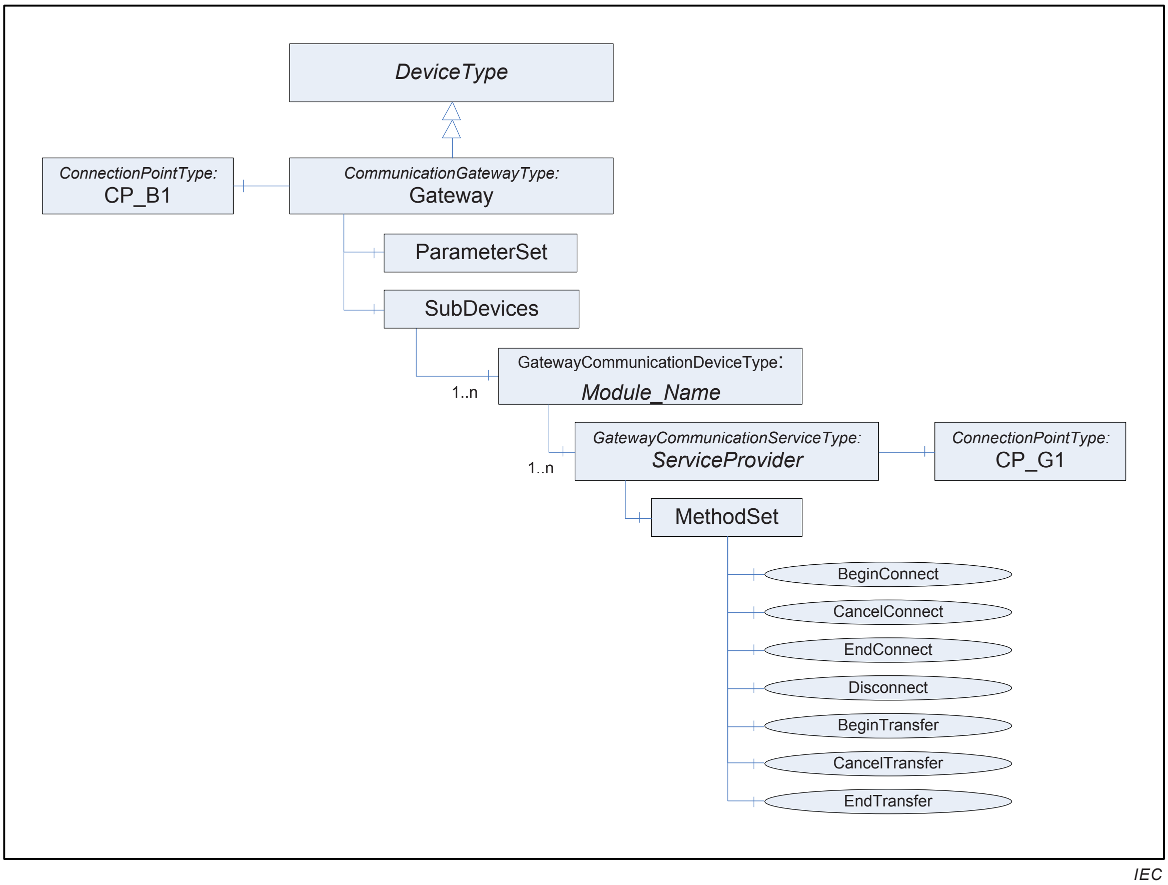


Figure 12 – Gateway information model

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 27.

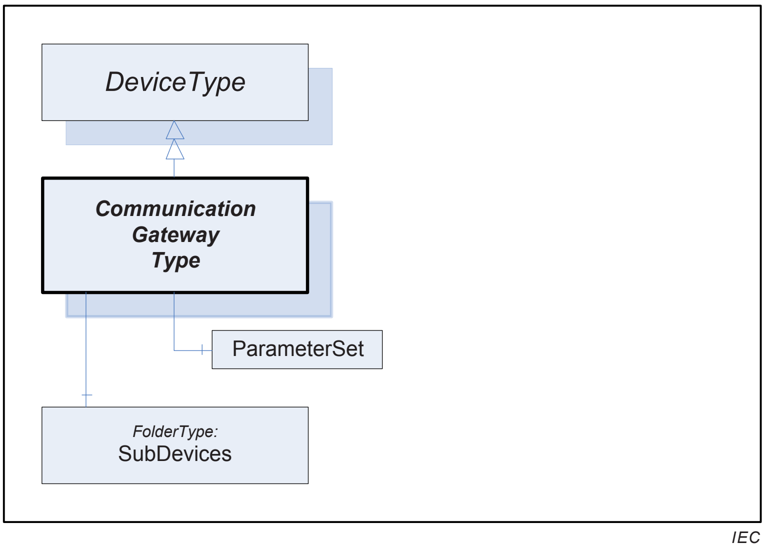


Figure 13 – CommunicationGatewayType

Table 27 – 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 62769-4:2015, Annex F. Figure 14 shows the GatewayCommunicationDeviceType definition that is formally defined in Table 28 and Table 29.

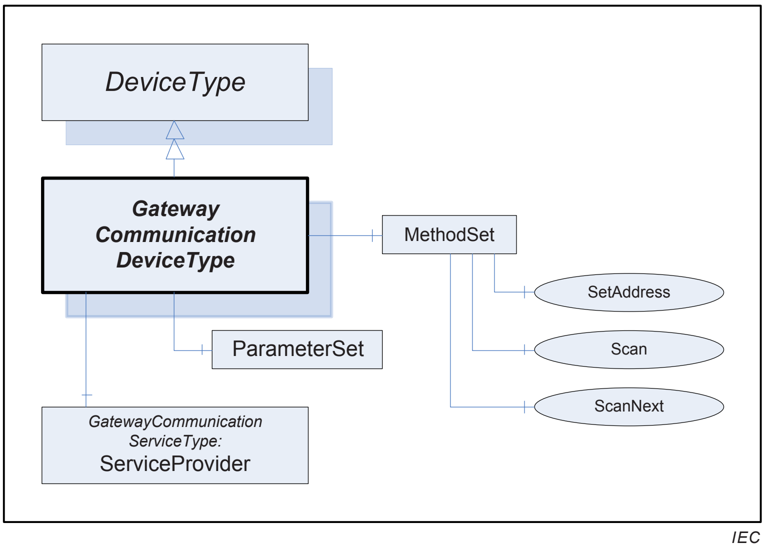


Figure 14 – GatewayCommunicationDeviceType

Table 28 – 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
HasComponent	Object	ParameterSet		BaseObjectType	Optional
HasComponent	Object	ServiceProvider		Gateway Communication ServiceType	Mandatory

Table 29– 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 enables 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 30 and Table 31 specify the arguments and AddressSpace representation, respectively.

Signature

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

Table 30 – 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 31 – 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 enables 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 32 and Table 33 specify the arguments and AddressSpace representation, respectively.

Signature

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

Table 32 – 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 33 – 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 8.3.2.4.

`<ListEntry>`: The attribute value shall refer to the COLLECTION definition that describes the Connection Point as defined in 8.3.2.4.

8.2.3.5 SetAddress Method

For definitions see 7.3.3.4.

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 methods 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.

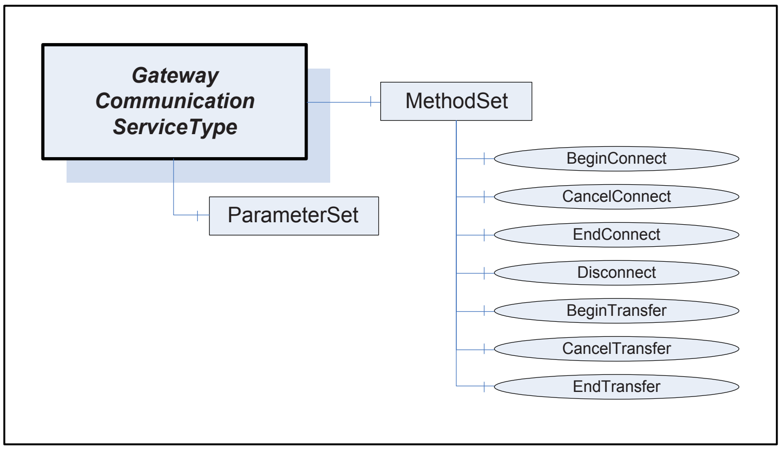


Figure 15 – GatewayCommunicationServiceType

Table 34 – 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 35 – 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 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 62769-4:2015, 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 36 specifies the arguments.

Signature

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

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

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

Table 36 – 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	A protocol specific argument list that is used for the address and optional device identification data (details are described in IEC 62769-4:2015, 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 37.

Signature

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

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

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

Table 37 – Transfer Method arguments

Argument	Description
CommunicationRelationId	See 8.2.4.2.
SendData	A protocol specific list of values as described in IEC 62769-4:2015, Annex F. The argument values represent the protocol specific communication service request that is sent to the device.
ReceiveData	A protocol specific list of values as described in IEC 62769-4:2015, 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 Built-in library function invocation or the writing of data onto an online node. The set of usable EDDL Built-in 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 35

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 behavior

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 device is connected to a PROFIBUS 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). The way the HART Command is wrapped into the PROFIBUS communication request may 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 either be standard or Gateway specific.

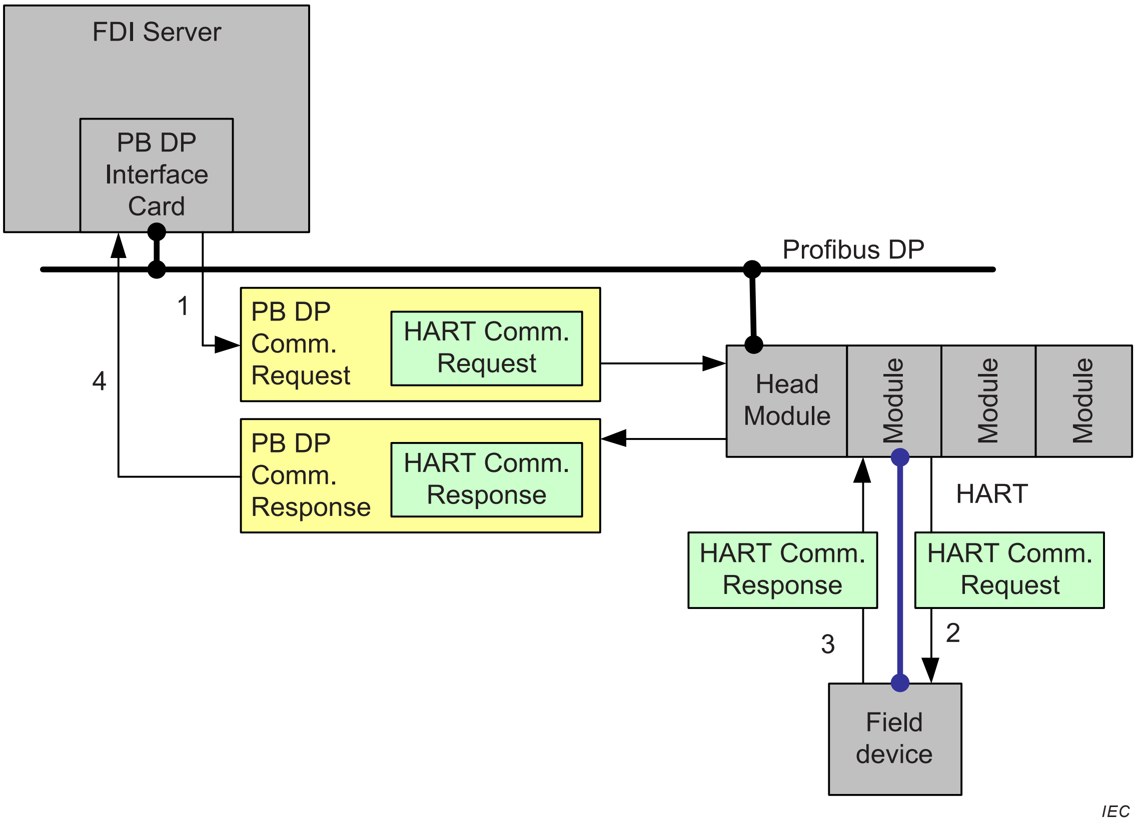


Figure 16 – Nested Communication

Subclause 8.4 defines the Gateway handling and behavior 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 behavior 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 behavior 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 62769-4:2015, 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 Built-in 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 device may 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 Connection Point

Since Connection Point description is based on EDDL an actual inheritance approach known from object oriented programming languages doesn't 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. The following shows some EDDL examples of how Connection Point declarations follow this naming convention.

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

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 IM

Connection Points sharing a certain set of address formation may 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 appendix defines the numeric identifiers for all of the numeric *NodeIds* defined in this standard. The identifiers are specified in a CSV file with the following syntax:

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

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

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

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

The CSV released with this version of the standard can be found here:
http://www.fdi-cooperation.com/tl_files/Specification/1.0/Schemas/Opc.Ua.Fdi7.NodeIds.csv

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

The Information Model Schema released with this version of the standard can be found here:
http://www.fdi-cooperation.com/tl_files/Specification/1.0/Schemas/Opc.Ua.Fdi7.NodeSet2.xml

Bibliography

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

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

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

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

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

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

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

____

This page deliberately left blank

This page deliberately left blank

British Standards Institution (BSI)

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

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

About us

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

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

Information on standards

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

Buying standards

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

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

Subscriptions

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

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

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

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

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

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

Revisions

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

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

Copyright

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

Useful Contacts:

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

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

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

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

BSI Group Headquarters

389 Chiswick High Road London W4 4AL UK

bsi.

...making excellence a habit.™