BS EN 62769-2:2015

BSI Standards Publication

Field Device Integration (FDI)

Part 2: FDI Client

bsi.

...making excellence a habit.™

National foreword

This British Standard is the UK implementation of EN 62769-2:2015. It is identical to IEC 62769-2:2015.

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

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

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

ISBN 978 0 580 78324 1
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 62769-2

June 2015

ICS 25.040.40; 35.100

English Version

Field Device Integration (FDI) - Part 2: FDI Client (IEC 62769-2:2015)

Intégration des appareils de terrain (FDI) - Partie 2: Client FDI
(IEC 62769-2:2015)

Feldgeräteintegration (FDI) - Teil 2: FDI-Client
(IEC 62769-2:2015)

CENELEC

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

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

European foreword

The text of document 65E/345/CDV, future edition 1 of IEC 62769-2, 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 62769-2:2015.

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• latest date by which the national standards conflicting with the document have to be withdrawn	(dow)	2018-06-16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 62769-2:2015 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 62541-3	-	OPC unified architecture - Part 3: Address Space Model	EN 62541-3	-
IEC 62541-4	-	OPC Unified Architecture - Part 4: Services	EN 62541-4	-
IEC 62769-1	-	Devices and integration in enterprise systems; Field Device Integration - Part 1: Overview	-	-
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 62769-6	2015	Devices and integration in enterprise systems; Field Device Integration - Part6: Technology Mapping	-	-
ISO 639	-	Code for the representation of names of languages	-	-
ISO 3166	-	Codes for the representation of names of countries	-	-
ISO/IEC 10918-1	-	Information technology; digital compression and coding of continuous-tone still images; requirements and guidelines	-	-
ISO/IEC 15948	-	Information technology - Computer graphics and image processing - Portable Network Graphics (PNG) - Functional specification	-	-
IEEE 754	-	IEEE Standard for Binary Floating-Point Arithmetic	-	-
IETF RFC 2083	-	PNG (Portable Network Graphics) - Specification Version 1.0	-	-
IETF RFC 3066	-	Tags for the Identification of Languages	-	-
XML-1	-	XML Schema Part 1: Structures, W3C	-	-
XML-2	-	XML Schema Part 2: Datatypes, W3C	-	-

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

FIELD DEVICE INTEGRATION (FDI) –

Part 2: FDI Client

FOREWORD

International Standard IEC 62769-2 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/345/CDV	65E/422/RVC

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

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

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

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

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

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

INTRODUCTION

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

a) Method for the supplying and installation of device-specific functionalities, see Patent Family DE10357276;
b) Method and device for accessing a functional module of automation system, see Patent Family EP2182418;
c) Methods and apparatus to reduce memory requirements for process control system software applications, see Patent Family US2013232186;
d) extensible device object model, see Patent Family US12/893,680.

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

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

a) ABB Research Ltd
Claes 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 2: FDI Client

1 Scope

This part of IEC 62769 specifies the FDI Client. The overall FDI architecture is illustrated in Figure 1. The architectural components that are within the scope of this document have been highlighted in this figure.

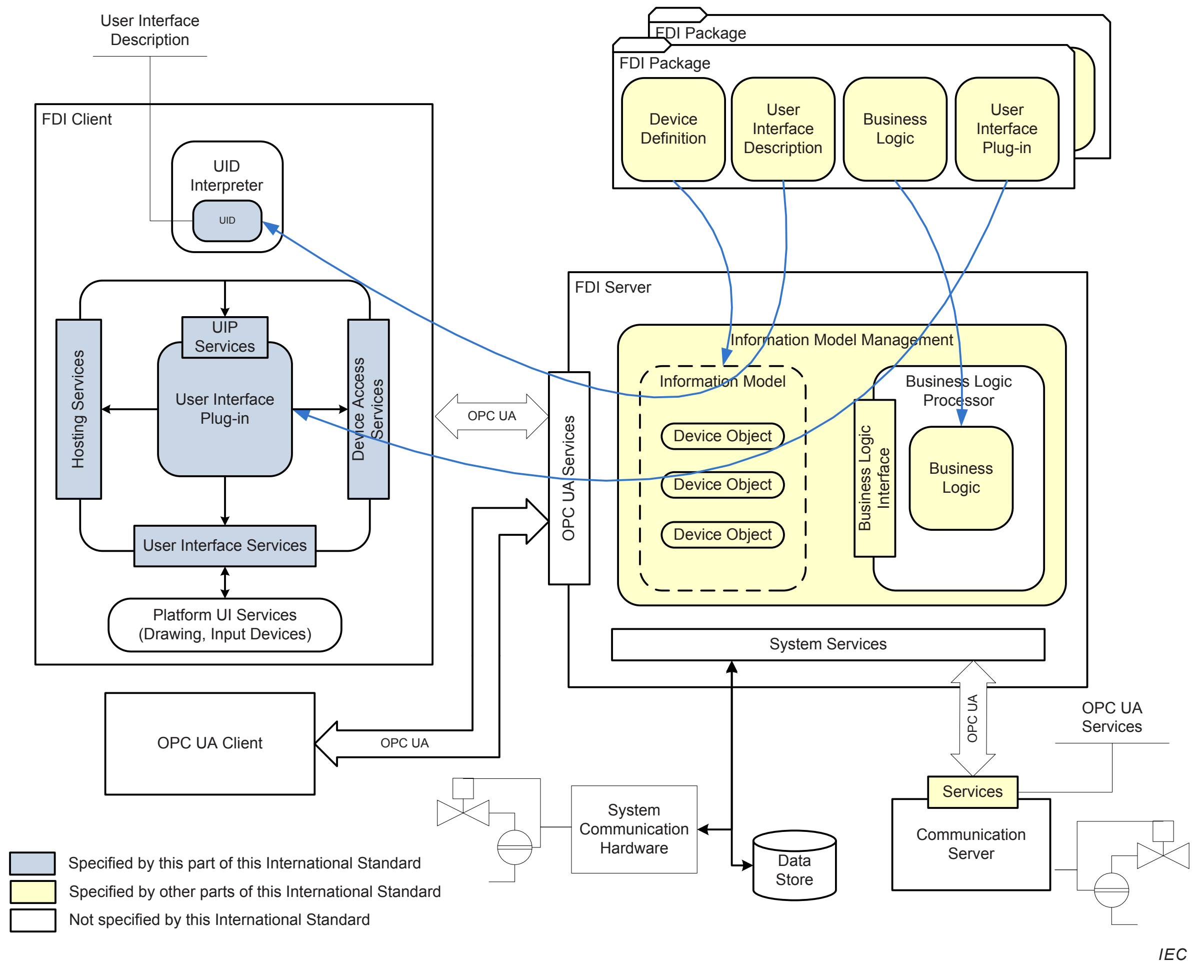


Figure 1 – FDI architecture diagram

2 Normative references

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

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

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

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.

IEC 62769-6:2015, *Field Device Integration (FDI) – Part 6: FDI Technology Mapping*

NOTE IEC 62769-6 is technically identical to FDI-2026.

IEC 62541-3, *OPC Unified Architecture – Part 3: Address Space Model*

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

ISO 639, *Codes for the representation of names of languages*

ISO 3166, *Codes for the representation of names of countries and their subdivisions*

ISO/IEC 10918-1, *Information technology – Digital compression and coding of continuous-tone still images: Requirements and guidelines*

ISO/IEC 15948, *Information technology – Computer graphics and image processing – Portable Network Graphics (PNG): Functional specification*

IEEE 754, *IEEE Standard for Floating-Point Arithmetic*

IETF RFC 2083, *PNG (Portable Network Graphics) Specification Version 1.0*

IETF RFC 3066, *Tags for the Identification of Languages*

XML Schema-1, *XML Schema: Structures* (available at http://www.w3.org/TR/xmlschema-1/)

XML Schema-2, *XML Schema: Datatypes* (available at http://www.w3.org/TR/xmlschema-2/)

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 Terms used for Services

3.1.1.1
Locking Services
set of Services through which access to a Device is controlled

3.1.1.2
Device Model Services
sub-set of the Device Access Services through which a UIP can access the information of a Device

3.1.1.3
Direct Access Services
sub-set of the Device Access Services through which a UIP can directly access a Device

3.1.2 Terms used for Device Access Services

3.1.2.1
Attribute
information element of a Node

Note 1 to entry: Some Attributes exist for all NodeClasses and some are specific to a given NodeClass.

Note 2 to entry: Supersedes the definition given in IEC 62769-1.

3.1.2.2
Device Model
hierarchy of Nodes that represents an existing Device

3.1.2.3
Node
element in the Device Model that can be addressed via the Device Access Services

Note 1 to entry: Supersedes the definition given in IEC 62769-1.

3.1.2.4
NodeClass
either an Object or a Variable

Note 1 to entry: Supersedes the definition given in IEC 62769-1.

3.1.2.5
Object
instance of the Object NodeClass

Note 1 to entry: Supersedes the definition given in IEC 62769-1.

3.1.2.6
Variable
instance of the Variable NodeClass

Note 1 to entry: Supersedes the definition given in IEC 62769-1.

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.

UTC	Coordinated Universal Time
XML	Extended mark-up language

3.3 Conventions

Conventions for service definitions are identical to IEC 62541-4.

Basic data types used are defined in IEC 62541-3.

“Parameter" is always an Information Model Parameter. "parameter" is the general use of the word. If ambiguous, additional context is given.

4 Overview

An FDI Package provides the necessary information for a Device Type to allow managing the Device within the system. It is provided by a device vendor and deployed in an FDI Server. It may contain two types of user interface components that are available to the FDI Client for display to a user. An FDI Package may contain only one type or both types. The two types are referred to as User Interface Plug-ins and User Interface Descriptions.

A User Interface Plug-in (UIP) is an executable element. A UIP is provided by an FDI Package and transferred to the FDI Client by the FDI Server. A UIP provides a set of UIP Services that the FDI Client uses to initialize and interact with the UIP.

NOTE 1 IEC 62769-6 defines application programming interfaces for the services described in this document.

User Interface Descriptions (UIDs) are defined using EDDL. A UID is provided to the FDI Client by the FDI Server. The FDI Client uses the UID Interpreter to interpret and execute the UID. A UID may make use of other UID and UIP components as subcomponents in order to provide a modular approach and make the best use of both descriptive and executable user interface elements.

NOTE 2 UIPs can make use of other UIPs but not of other UIDs.

The FDI Server makes UIDs and UIPs available to the FDI Client via the Information Model. The Information Model organizes the UIDs and UIPs by Device Type.

The FDI Client provides the execution environment for UIPs. The FDI Client loads the UIP from the FDI Server.

The FDI Client’s UIP execution environment consists of the following sets of services that are made available to the UIP:

- Device Access Services
- Hosting Services
- User Interface Services
- Printing Services (if available in the hosting environment)

NOTE 3 It is implementation specific whether different UIPs get the same or different interface instances to access these services. The only requirement is that there is no side-effect if two UIPs use the same instance of an interface.

Similar to the FDI Client, each UIP shall also provide a set of services (UIP Services) by which the FDI Client activates, controls and shuts down UIPs (see 6.1).

The Device Access Services enable interaction between the UIP and the FDI Server-maintained Information Model. The FDI Client takes care of the interaction with the FDI Server freeing the UIP to focus on the application level only.

UIP access to the Information Model (IM) via the Device Access Services is restricted to the Device and its sub-devices.

The Hosting Services are provided by the FDI Client for use by the UIP. The Hosting Services include services related to the FDI Client allowing the UIP to acquire information about the environment.

The User Interface Services provide the means by which the UIP accesses the user interface services of the underlying operating system. These services provide access to the screen, keyboard, mouse, and other operating system resources. The User Interface Services are defined by the chosen implementation technology and therefore no additional definitions are included in this document (see IEC 62769-6). UIPs shall use the Hosting Services to display message boxes or progress bars and shall never use comparable services provided by the underlying operating system.

There are no printing services provided by the Client. If a UIP needs to generate a printout it accesses the printing services of the underlying operating system. No additional definitions are included in this document.

The FDI Client uses the culture setting of the currently signed-in user for the execution environment of the UIP. It uses the culture when creating OPC UA Sessions and it sets the culture for each thread it creates. UIPs shall take care for culture setting in all threads that they create.

The FDI Client provides a UID Interpreter that is used to interpret and execute UIDs. The UID XML Schema is defined in this document (see Annex A).

Business Logic is executed in the FDI Server. Some Business Logic may be exposed to the FDI Client as Actions (see Clause 7) and can be triggered by FDI Clients.

5 FDI Client

5.1 Device Access Services

5.1.1 General

The Device Access Services provide access to both the online and offline information of a Device or its components as defined by the FDI Package, in particular for

- browsing the Device Model,
- reading / writing of data and subscribing to data changes,
- controlling access to the Device, and
- directly communicating with the Device.

The scope for the Device Access Services will be a Device, Block, or Communication Server to which the UIP is assigned. The FDI Client is expected to map the Device Access Services to OPC UA services provided by the FDI Server.

The main Services are the Device Model Services to view and access Parameters. The Locking Services are used to control simultaneous access to a Device. The Direct Access Services allow a UIP to communicate with the Device.

Whether and how the different services are mapped to real interfaces is defined in IEC 62769-6. IEC 62769-6 also specifies how interfaces are obtained.

5.1.2 Device Model

The Device Model defines the structure of all data that are available to a UIP. It is confined to a single device instance. The entities in the structure (Parameters, Images, and Documents) are built from FDI Package information. User interface elements, like Menus, Graphs, Waveforms, are not part of the UIP Device Model.

All Device elements are organized as a defined hierarchy. The root of the hierarchy can be either a Device or a Block subject to where (by which MENU) the UIP is referenced in the User Interface Description of the FDI Package (see IEC 62769-4). The Nodes in the hierarchy are either Objects or Variables, where the main difference is that Variables provide a Value.

Figure 2 illustrates the overall structure of a Device. Figure 3 shows the structure of a block in more detail. The elements that will really be available depend on the contents of the respective FDI Package. Regular rectangles represent Object Nodes while the ones with rounded corners represent Variable Nodes. These NodeClasses are defined in 5.1.3. The top left Node is the root Node. The single hashed lines define the parent-child relationship in the hierarchy. As an example, the children of Device1 in Figure 2 are ParameterSet, ImageSet, Documentation, Blocks and SubDevices. The children of /SubDevices/Device_1b/ImageSet are Image_1 and Image_2.

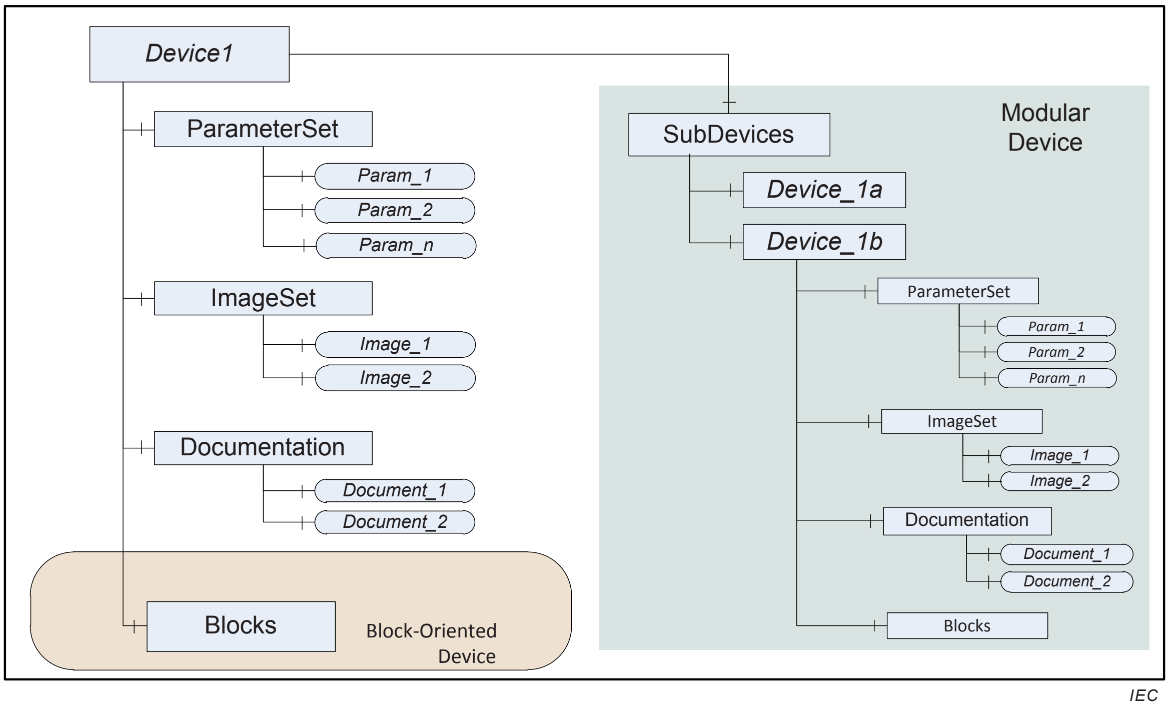


Figure 2 – Overall structure of a Device

Names that are in normal font are defined by the Device Access Services; names in italics are just place-holders for the real names as defined in the FDI Package.

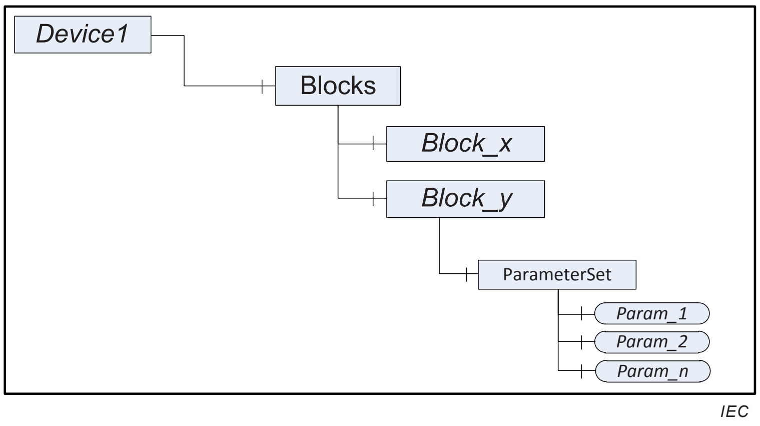


Figure 3 – Structure of Blocks

Each Node in the hierarchy is uniquely qualified with its pathname. This pathname is a concatenation of the individual Node Name Attributes. The separator is “/”.

EXAMPLE The following examples illustrate qualified pathnames:

/	-- the root Node (here “Device1”)
/ParameterSet/Param_2	-- a Variable Node
/SubDevices/Device_1b/ImageSet/Image_1	-- a picture

Certain Variables in the FDI Package may be tagged as “private” meaning they are non-browsable. Though they are non-browsable, they exist in the Device Model and can be addressed with their pathname.

5.1.3 Node model

5.1.3.1 General

The information of a Device is organised as a hierarchy of Nodes. Each Node is either an Object or a Variable. Both Object and Variable are derived from the BaseNodeClass, as illustrated in Figure 4.

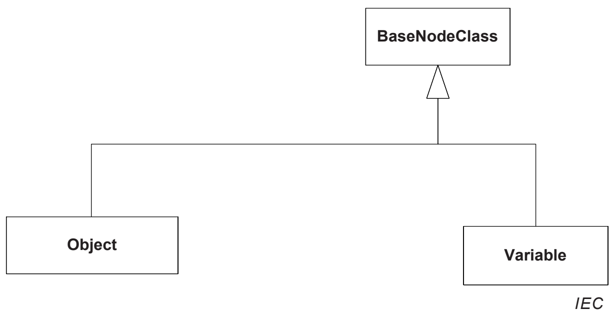


Figure 4 – Device Model NodeClasses

5.1.3.2 BaseNodeClass

This is the abstract parent NodeClass for Object and Variable Nodes. The BaseNodeClass Attributes are shown in Table 1. The Attributes of this NodeClass are available in both Object and Variable NodeClasses.

Table 1 – BaseNodeClass Attributes

Attribute	Datatype	Description
NodePath	String	Qualified path of the Node in the Device Model. This Attribute is returned by the Browse Service and cannot be read or written.
Name	String	Name of Node according to device specific documentation.
Label	LocalizedText	Human readable label of the Node.
Description (optional)	LocalizedText	Human readable help string describing the Node.

Additional Attributes will be available for derived NodeClasses. For example, a Variable will have Attributes that define the DataType and the AccessRights.

5.1.3.3 Object NodeClass

Table 2 contains the list of Attributes for Objects beyond those inherited from the Base NodeClass:

Table 2 – Object NodeClass Attributes

Attribute	Datatype	Description
LockedStatus	Boolean	This Attribute when "true" indicates that this Object is currently locked. Bad_AttributeInvalid defines that locking is not supported for this Object at all.

5.1.3.4 Variable NodeClass

5.1.3.4.1 General

Variables are used to represent Parameters, images and documents. When reading Variables that represent images or documents the system will provide them as a ByteString. For documents, the Name Attribute will consist of the filename including the extension that can be used to identify the document type. FDI supports ".pdf" and ".txt". For the representation of images see 5.1.3.4.2.

Table 3 contains the list of Attributes for Variables beyond those inherited from the BaseNodeClass:

Table 3 – Variable NodeClass Attributes

Attribute	Datatype	Description
Value	Variant	The value of the Variable as returned from the device (i.e. without applying the ScalingFactor). The Variant is specified in 5.1.9.4.
DataType	UInt32	The DataType Attribute specifies the data type of the Value Attribute. One of the data types specified in 5.1.9. IEC 62769-6 specifies the assignment of unique identifiers to each type.
ValueRank	Int32	Indicates whether the Value Attribute is an array. It may have the following values: >1 (MoreDimensions) – the value is an array with the specified number of dimensions. 1 (OneDimension) – the value is an array with one dimension. 0 (OneOrMoreDimensions) – the value is an array with one or more dimensions. -1 (Scalar) – the value is not an array. -2 (Any) – the value can be a scalar or an array with any number of dimensions. -3 (ScalarOrOneDimension) – the value can be a scalar or a one dimensional array.
ArrayDimensions (optional)	UInt32[]	Specifies the length of each dimension for an array value. The Attribute is intended to describe the capability of the Variable, not the current size. The number of elements shall be equal to the value of the ValueRank Attribute. Shall be null if ValueRank <= 0. A value of 0 for an individual dimension indicates that the dimension has a Variable length. For example, if a Variable is defined by the following C array: Int32 myArray[346]; then this Variable's DataType would point to an Int32, the Variable's ValueRank has the value 1 and the ArrayDimensions is an array with one entry having the value 346.
AccessRights	Byte	The access rights for the Value. An enumeration with one of the following values: NONE_0 The Variable value cannot be accessed READ_1 The value of the Variable may be read WRITE_2 The value of the Variable may be written READORWRITE_3 The value of the Variable may be read or written
UserAccessRights	Byte	This Attribute specifies the access rights to the Value for the currently authenticated user. They may be less than the potential access rights. The same enumeration as for AccessRights is used.
ScalingFactor (optional)	Double	This Attribute specifies a suggested scaling factor. Note that the Value Attribute contains the raw value returned from the device. It is assumed, that the (raw) value is multiplied by this factor before being displayed.
EngineeringUnits (optional)	EUInformation	EngineeringUnits specifies the units for the value (e.g., °C, hertz, seconds). See 5.1.9.3.8 for the definition of the EUInformation data type.
Attributes for analog Variables (Variables that represent continuously-variable physical quantities (e.g., pressure, temperature)).		
EURange (optional)	Range[]	Defines one or more value ranges likely to be obtained in normal operation. They are intended for such use as automatically scaling a bar graph display. Sensor or instrument failure or deactivation can result in a returned item value that

Attribute	Datatype	Description
		is actually outside this range. UIP software shall be prepared to deal with this. See 5.1.9.3.7 for the definition of the Range data type. Ranges may change during operation , for example by changing the operation mode of an instrument. Like the Value itself, Ranges are always unscaled (i.e. without applying the ScalingFactor).
Attributes for discrete (enumerated) Variables (for data that may take on only a certain number of possible values (e.g., OPENING, OPEN, CLOSING, CLOSED).		
CurrentLabel	String	Enumerated Variables expose the current numeric state in their Value Attribute. The CurrentLabel Attribute provides the name of the current enumeration value.
EnumValues	EnumValuesType[]	EnumValues is an array of {StateValue, Enumeration Name, and Help Information}. See 5.1.9.3.9 for the definition of this type. FDI Clients/UIPs may read this Attribute in advance and store it for lookup of name or help when they receive the numeric representation.
Attributes for bit-enumerated Variables (for data that represent a bit mask).		
OptionNames	String[]	Bit-enumerated Variables transmit a bit mask encoded in an unsigned integer of a length that is sufficient to represent all bits. The OptionNames Attribute provides a human-readable representation for each valid bit of the bit mask. The order of the bits of the bit mask points to a position of the array of Strings in the OptionNames Attribute, i.e., the first bit points to the first entry in the array, and so on. The array contains an empty String for each bit that has no specific meaning.

5.1.3.4.2 Representation of images

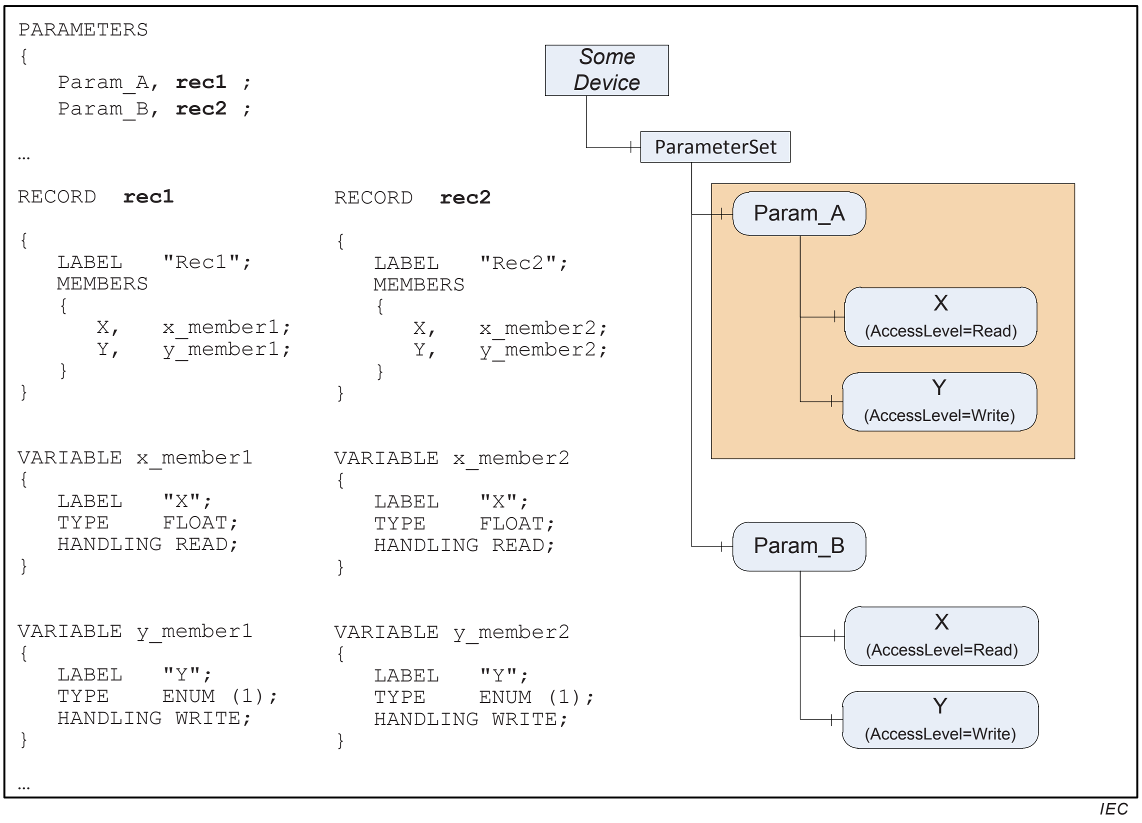
All images have the DataType and are transferred as a ByteString. FDI supports three image formats. To identify the format of the image provided in the ByteString, the initial bytes have to be parsed as outlined in the following table.

Image type	Description
GIF	Defines an image in GIF (Graphics Interchange Format). GIF is specified in http://www.w3.org/Graphics/GIF/spec-gif89a.txt . The first bytes of a GIF image are as follows: Byte | 1 | 2 | 3 Hex | 47 | 49 | 46
JPG	Defines an image in JPG (Joint Photographic Experts Group File Interchange Format). JPG is defined in ISO/IEC 10918-1. The first bytes of a JPG image are as follows: Byte | 1 | 2 | 3 | 4 Hex | FF | D8 | FF | E0
PNG	Defines an image in PNG (Portable Network Graphics format). PNG is defined in IETF RFC 2083 and ISO/IEC 15948. The first bytes of a PNG image are as follows: Byte | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 Hex | 89 | 50 | 4E | 47 | 0D | 0A | 1A | 0A

5.1.3.4.3 Representation of records

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

An example of how a record is represented in the Device Model is shown in Figure 5.



```
PARAMETERS
{
   Param_A, rec1 ;
   Param_B, rec2 ;

…

RECORD  rec1                RECORD  rec2

{                           {
   LABEL   "Rec1";             LABEL   "Rec2";
   MEMBERS                     MEMBERS
   {                           {
      X,    x_member1;            X,    x_member2;
      Y,    y_member1;            Y,    y_member2;
   }                           }
}                           }


VARIABLE x_member1          VARIABLE x_member2
{                           {
   LABEL   "X";                LABEL   "X";
   TYPE    FLOAT;              TYPE    FLOAT;
   HANDLING READ;              HANDLING READ;
}                           }


VARIABLE y_member1          VARIABLE y_member2
{                           {
   LABEL   "Y";                LABEL   "Y";
   TYPE    ENUM (1);           TYPE    ENUM (1);
   HANDLING WRITE;             HANDLING WRITE;
}                           }

…
```

Figure 5 – Example: Variable hierarchy representing a RECORD

The Name and Label Attributes of the root Variable are set to the name of the RECORD and the LABEL Attribute, respectively. The DataType Attribute of the “root” Variable is Variant. The ValueRank Attribute is used to specify that the Value contains an array. The Value Attribute represents the values of all members in the order as defined for the RECORD. According to the example in Figure 5, the first variant will be a FLOAT and the second will be an ENUM.

For each component Variable that represents an EDDL RECORD MEMBER

- the Name Attribute is set to the identifier of the corresponding EDDL VARIABLE,
- the Label is the LABEL Attribute of the corresponding EDDL VARIABLE,
- the Description is the HELP Attribute of the corresponding EDDL VARIABLE, and
- the AccessRights Attribute is derived from the EDDL HANDLING Attribute.

Each member of the record can be accessed with its pathname as specified above. Browsing can also be used.

EXAMPLE Example of a pathname: /ParameterSet/Param_A/X.

5.1.3.4.4 Representation of arrays, and lists of members with simple data types

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

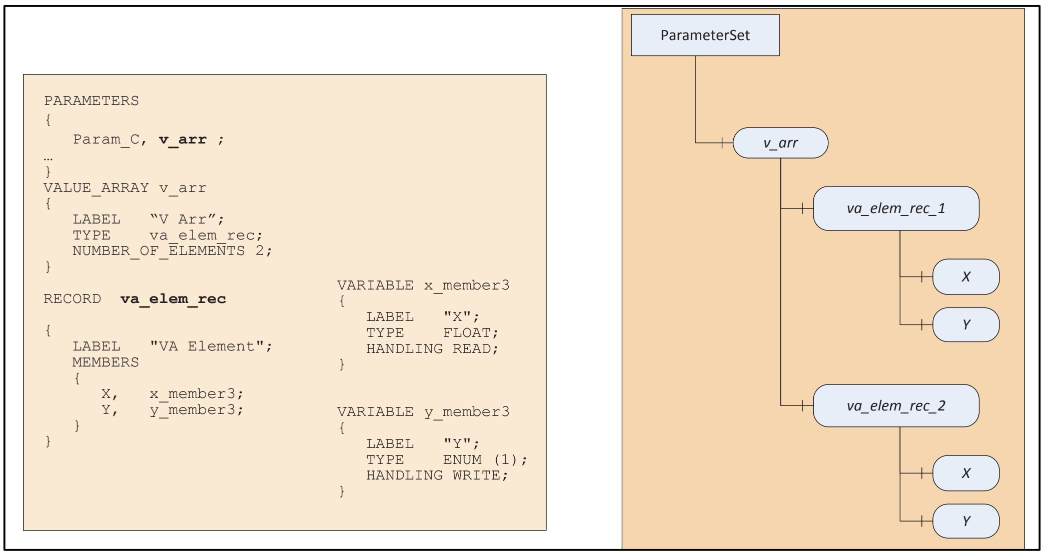
The DataType Attribute is set to one of the base data types (see 5.1.9.2).

The ValueRank Attribute is used to specify that the Value contains an array. In case of an EDDL VALUE_ARRAY the number of elements is exposed via the ArrayDimensions Attribute.

In the case of an EDDL LIST the number of elements is unspecified since the size can change dynamically.

5.1.3.4.5 Representation of arrays, and lists of RECORDs

Value arrays or lists of non-simple EDDL Parameters will be represented as an array of Variable hierarchies. Figure 6 shows the EDDL sample code of a VALUE_ARRAY of RECORDs and the corresponding Variable hierarchy.



```
PARAMETERS
{
   Param_C, v_arr ;
…
}
VALUE_ARRAY v_arr
{
   LABEL   "V Arr";
   TYPE    va_elem_rec;
   NUMBER_OF_ELEMENTS 2;
}
RECORD  va_elem_rec

{
   LABEL   "VA Element";
   MEMBERS
   {
      X,   x_member3;
      Y,   y_member3;
   }
}
```

```
VARIABLE x_member3
{
   LABEL   "X";
   TYPE    FLOAT;
   HANDLING READ;
}

VARIABLE y_member3
{
   LABEL   "Y";
   TYPE    ENUM (1);
   HANDLING WRITE;
}
```

Figure 6 – Variable hierarchy representing a VALUE_ARRAY of RECORDs

The Name and Label Attributes of the root Variable are set to the name of the ARRAY and the LABEL Attribute, respectively. The DataType Attribute of the "root" DataVariable is Variant. The ValueRank Attribute is used to specify that the Value contains an array. The Value Attribute represents all VALUE_ARRAY entries. The first Variant corresponds to the first array entry and so on. Each Variant in turn contains an array. This may either be an array of simple types or an array of Variants. A RECORD is always represented as an array of Variant.

The VALUE_ARRAY element, which is in fact a RECORD, is represented as a component Variable hierarchy. The Name and Label Attributes of each root Variable representing a RECORD are set to the name of the RECORD and the LABEL Attribute, respectively. The array index (_1, _2) is appended to allow unique identification. Note that the index always begins with '1'.

The RECORD MEMBERS are also represented as component Variables as specified in 5.1.3.4.3.

Members of each record can be accessed with a pathname. Browsing can also be used.

EXAMPLE Example of a pathname: /ParameterSet/v_arr/va_elem_rec_2/Y.

5.1.4 Services

5.1.4.1 General

All Services specified in this part of IEC 62769 rely on the conventions defined in 5.1.4.2. They are specified in an abstract manner with request and response parameters in a single table.

Programmatic access to the services may be synchronous or asynchronous (see IEC 62769-6). However, asynchronicity exists to maintain responsiveness of the User Interface. Service execution shall always be assumed to be sequential.

5.1.4.2 Conventions for service definitions

The service specifications use tables to describe service parameters, as shown in Table 4. Parameters are organised in this table into request parameters and response parameters.

Table 4 – Service Definition Table

Name	Type	Description
Request		Defines the request parameters of the service
simple Parameter Name		Description of this parameter
constructed Parameter Name	structure	Description of the constructed parameter
component Parameter Name		Description of the component parameter
Response		Defines the response parameters of the service

The Name, Type and Description columns contain the name, data type and description of each parameter. All parameters are mandatory, although some may be unused under certain circumstances. The description column specifies the value to be supplied when a parameter is unused. Parameter names always begin with a lower case character. This allows differentiating if name and type are the same, for example, name = “nodeId”, type = “NodeId”.

Two types of parameters are defined in these tables, simple and constructed. Simple parameters have a simple data type, such as Boolean or String.

Constructed parameters are composed of two or more component parameters, which can be simple or constructed. Component parameter names are indented below the constructed parameter name.

The data types used in these tables may be base types or service-specific types. Base data types are listed in 5.1.9. Data types that are service-specific are defined in the parameter table of the service.

5.1.4.3 Auditing

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

When an audit trail is maintained by the system, audit trail records for the Write Service invoked by the UIP will be implicitly created by the system.

In addition, UIPs have means for providing additional audit context information for things that the system cannot know:

- They can call the LogAuditTrailMessage service (see 5.2.2.5). This shall be executed in particular when the DirectAccess Services are used. The UIP shall include sufficient information in the message for precise description of the activity performed.
- They can provide a context text when using the Locking Services (see 5.1.7) or the Direct Access Services (see 5.1.8).

5.1.4.4 Services and operations

Several of the Device Model Services (e.g. Read and Write) allow specifying an array of elements that shall be processed. The processing of each individual element is called an "operation". This is an important differentiation for error handling as a service execution is considered successful even if individual operations fail. The following list explains the differences between the service and operation result codes.

- Service result code

  If a service succeeds, the result code is "Good" and the response parameters are valid. If it fails, the service result code is any of the "Bad_..." service failure codes defined in 5.1.4.6. In such a case an InnerErrorInfo (see 5.1.9.3.4) may be provided as well. Programmatic access to service failure codes is specific to the technology and may be based on exceptions (see IEC 62769-6).
- Operation result code

  The result code for each operation is returned as part of the service-specific response parameters (see 5.1.4.7 for the list of available operation result codes).

Operations can return an InnerErrorInfo with each result code other than "Good" if returnInnerErrorInfo="true" in the service request.

5.1.4.5 StatusCode

The StatusCode used for service results or operational results reports the outcome of an operation. It is a 32-bit unsigned integer. The top 16 bits represent the numeric value of the code that shall be used for detecting specific errors or conditions. The bottom 16 bits are bit flags that contain additional information but do not affect the meaning of the StatusCode.

UIPs shall always check the StatusCode associated with a result before using it. Results that have an uncertain/warning status associated with them shall be used with care since these results might not be valid in all situations. Results with a bad/failed status shall never be used.

The exact bit assignments are shown in Table 5. IEC 62769-6 provides functions to help in evaluation of the StatusCode.

Table 5 – StatusCode Bit Assignments

Field	Bit Range	Description		
Severity	30 .. 31	Indicates whether the StatusCode represents a good, bad or uncertain condition. These bits have the following meanings:		
		Severity	**Bits**	**Description**
		Good Success	00	The operation was successful; results may be used.
		Uncertain Warning	01	The operation was partially successful; results might not be suitable for some purposes.
		Bad Failure	10	The operation failed and any associated results cannot be used.
		Reserved	11	Reserved for future use. Should also be treated as "Bad".
Reserved	29 .. 28	Reserved for future use. Shall always be zero.		
SubCode	16 .. 27	The code is a numeric value assigned to represent different conditions. Each code has a symbolic name and a numeric value. All descriptions in this specification refer to the symbolic name. IEC 62769-6 maps the symbolic names onto a numeric value.		
Reserved	12 .. 15	Reserved for future use. Shall always be zero.		
InfoType	10 .. 11	The type of information contained in the info bits. These bits have the following meanings:		
		InfoType	**Bits**	**Description**
		NotUsed	00	The info bits are not used and shall be set to zero.
		DataValue	01	The StatusCode and its info bits are associated with a data value returned from the FDI Server.
		Reserved	1X	Reserved for future use. The info bits shall be ignored.
InfoBits	0 .. 9	Additional information bits that depend on the Info Type field.		

Table 6 describes the structure of the InfoBits when the Info Type is set to DataValue (01).

Table 6 – DataValue InfoBits

Info Type	Bit Range	Description
LimitBits	8 .. 9	The limit bits associated with the data value. The limits bits have the following meanings:
Overflow	7	If this bit is set, not every detected change has been returned since the FDI Server's queue buffer for the subscribed Variable reached its limit and had to purge out data.
Reserved	0 .. 6	Reserved for future use. Shall always be zero.

Limit	Bits	Description
None	00	The value is free to change.
Low	01	The value is at the lower limit for the data source.
High	10	The value is at the higher limit for the data source.
Constant	11	The value is constant and cannot change.

5.1.4.6 Service failure codes

Table 7 defines result codes for the services. The column Service in Table 7 lists which code may be returned by which service. Result codes that are specific to an individual service are also specified with the service.

Table 7 – Service result codes

Exception name code	Description	Service
Bad_AlreadyLocked	The Node is already locked by another FDI Client.	InitLock
Bad_LockRequired	The passed Node is not yet locked.	Write, DirectAccess
Bad_MaxAgeInvalid	The max age parameter is invalid.	Read
Bad_NodeInvalid	The identifier does not refer to a valid Node in the Device Model. This result code is used both as service- and as operation-level result code.	Browse, InitLock, ExitLock
Bad_NothingToDo	There was nothing to do because the caller passed an empty list of operations.	Read , Write, Subscribe, Unsubscribe
Bad_NotSupported	The Service is not supported for the specified Node or for the Device/Block in general.	Locking, DirectAccess
Bad_RequestCancelled	The request was cancelled by Client / UIP.	All
Bad_SubscriptionIdInvalid	The subscription id is not valid.	Subscribe, Unsubscribe, DeleteSubscription
Bad_Timeout	The operation timed out.	All
Bad_TooManySubscriptions	The FDI Server has reached its maximum number of subscriptions.	CreateSubscription
Bad_InvalidState	The specified Node is in a state that does not permit this operation.	Services for Locking
Bad_TooManyOperations	The request could not be processed because it specified too many operations.	Read, Write, Subscribe
Bad_UnexpectedError	An unexpected error occurred.	All

5.1.4.7 Operational status codes

Table 8 defines the status codes for all operation level results (for services that have individual operations such as Read, Write or Subscribe). Each service defines the applicable subset and instead of a description will contain references to this table.

NOTE The value in Table 8 is referring to the Parameter value, not to the communication quality.

Table 8 – Operation level result codes

Operation Level Result Code	Description
Good	The operation completed successfully.
Good_LocalOverride	The value has been overridden.
Good_PostActionFailed	The value of a Variable was successfully read or written but one of the post actions failed.
Good_Edited	The Value has been modified in the EditContext and is not yet in the Device.
Good_DependentValueChanged	The Value of a dependent Variable was changed but not yet applied.
Uncertain	The operation completed however its outputs may not be usable.
Uncertain_NoCommunicationLastValue	Communication to the data source has failed. The Variable value is the last value that had a good quality.
Uncertain_LastUsableValue	Whatever was updating this value will no longer be doing so.
Uncertain_SubstituteValue	The value is an operational value that was manually overwritten.
Uncertain_InitialValue	The value is an initial value for a Variable that normally receives its value from another Variable.
Uncertain_SensorNotAccurate	The value is at one of the sensor limits.
Uncertain_EngineeringUnitsExceeded	The value is outside of the range of values defined for this parameter.
Uncertain_SubNormal	The value is derived from multiple sources and has less than the required number of good sources.
Uncertain_DominantValueChanged	A change of a dominant value – e.g. an engineering unit – made a dependent value invalid.
Bad	The operation failed.
Bad_AttributeInvalid	The attributeId is not supported for the specified Node.
Bad_ConfigurationError	There is a problem with the configuration that affects the usefulness of the value.
Bad_DeviceFailure	There has been a failure in the device/data source that generates the value.
Bad_IndexRangeInvalid	The indexRange parameter has an invalid syntax.
Bad_NodeInvalid	The identifier does not refer to a valid Node in the Device Model. This result code is used both as service- and as operation-level result code.
Bad_NotConnected	The Variable should receive its value from another Variable, but has never been configured to do so.
Bad_NotReadable	The access level does not allow reading or subscribing to the Node.
Bad_NotWritable	The access level does not allow writing to the Node.
Bad_OutOfRange	The value was out of range.
Bad_OutOfService	The source of the data is not operational.
Bad_SensorFailure	There has been a failure in the sensor from which the value is derived by the device/data source.
Bad_TypeMismatch	The value supplied is not of the same type as the Variable's value.
Bad_UIPHandleInvalid	The handle does not refer to a subscribed Node Attribute.
Bad_UserAccessDenied	User does not have permission to perform the requested operation.
Bad_WaitingForInitialData	Waiting for the FDI Server to obtain values from the underlying data source. After subscribing to Variables, it may take some time before values are delivered. In such cases an initial update may be sent with this status prior to the Notification with the first valid value.

5.1.5 Base Property Services

5.1.5.1 Overview

The Base Property Services provide access to basic Device Access properties.

5.1.5.2 Get DeviceAccess interface version

5.1.5.2.1 Description

This service returns the version of the interface that the UIP is using.

5.1.5.2.2 Parameters

Table 9 defines the parameters for the service.

Table 9 – GetDeviceAccessInterfaceVersion Service parameters

Name	Type	Description
Request		
Response		
version	String	FDI Technology Version of the DeviceAccess interface. The format of the value is xx.yy.zz as defined in IEC 62769-4.

5.1.5.2.3 Service results

There are no service results other than the common codes specified in 5.1.4.6.

5.1.5.3 GetOnlineAccessAvailability

5.1.5.3.1 Description

This service returns a hint as to whether online access is available in principle. This is general information. It does not clarify whether online access to a specific device is possible.

5.1.5.3.2 Parameters

Table 10 defines the parameters for the service.

Table 10 – GetOnlineAccessAvailability Service parameters

Name	Type	Description
Request		
Response		
onlineAccess	Boolean	This value when “false” specifies that online access is not available. “true”indicates basic availability.

5.1.5.3.3 Service results

There are no service results other than the common codes specified in 5.1.4.6.

5.1.6 Device Model Services

5.1.6.1 Overview

The Device Model Services include:

- Browse
- Read
- Write
- Subscription
  - CreateSubscription
  - Subscribe
  - Unsubscribe
  - DeleteSubscription

The services provide access to the offline representation and the online representation of the Device. The online Nodes for a Device are always present. However, access to data that require communication may be rejected with proper status codes such as Bad_NotConnected.

NOTE The online model does not contain SubDevices. If the UIP uses a path that includes SubDevices to access an online Node, this path will be evaluated in the offline hierarchy.

Cancel services are available for Browse, Read, and Write. The exact definition and mechanics depend on the technology used for the implementation of these services.

5.1.6.2 Browse

5.1.6.2.1 Description

Browses a single level in the hierarchy in the Device Model and returns Attributes for all children of the specified Node.

5.1.6.2.2 Parameters

Table 11 defines the parameters for the service.

Table 11 – Browse Service parameters

Name	Type	Description
Request		
nodeToBrowse	NodeSpecifier	The identifier of the Node to be browsed. See 5.1.9.3.2 for the definition of the NodeSpecifier type.
Response		
browseResult[]	structure	List of Nodes that are on the next level down in the hierarchy. For each Node the following Attributes will be returned.
nodePath	String	See BaseNodeClass in 5.1.3.2.
name	String	See BaseNodeClass in 5.1.3.2.
label	LocalizedText	See BaseNodeClass in 5.1.3.2.

5.1.6.2.3 Service results

No specific Service result codes are defined for Browse. Common StatusCodes are defined in Table 7.

5.1.6.3 CancelBrowse

5.1.6.3.1 Description

Calling CancelBrowse indicates that the UIP has no further interest in the results of this service. Execution will be stopped when possible.

Cancel is a suggestion to the system. Due to asynchronous execution, the service may already be fully or partially completed.

5.1.6.3.2 Parameters

Table 12 defines the parameters for the service.

Table 12 – CancelBrowse Service parameters

Name	Type	Description
Request		
serviceId	<technology dependent>	The identifier of the service to cancel.
Response		

5.1.6.3.3 Service results

No specific Service result codes are defined for CancelBrowse. Successfully cancelled Browse requests shall respond with Bad_RequestCancelled. Common StatusCodes are defined in Table 7.

5.1.6.4 Read

5.1.6.4.1 Description

This service is used to read Attributes of Object or Variable Nodes. UIPs that need to monitor Variable Attributes for changes shall use the Subscription services instead of Read.

5.1.6.4.2 Parameters

Table 13 defines the parameters for the service.

Table 13 – Read Service parameters

Name	Type	Description
Request		
returnInnerErrorInfo	Boolean	A value of “true”requests error information from calls to an underlying system to be returned when available. “false”defines that this information shall not be returned.
attributesToRead[]	Structure	The Attributes to read.
node	NodeSpecifier	The identifier of the Node that contains the Attribute to read. See 5.1.9.3.2 for the definition of the NodeSpecifier type.
attributeId	AttributeId	Numeric identifier of the Attribute to Read. See 5.1.9.3.3.
indexRange	NumericRange	This parameter is used to identify a single element of an array, or a single range of indexes for arrays. If a range of elements is specified, the values are returned as a composite. The first element is identified by index 0 (zero). This parameter is ignored if the specified Attribute is not an array or a structure. However, if the specified Attribute is an array or a structure, and this parameter is null (Null or empty String), then all elements are to be included in the range. See 5.1.9.3.6 for a detailed definition.
maxAge	UInt32	Maximum age of the value to be read in milliseconds. If the FDI Server has a cached value no older than maxAge, it will return the cached value rather than requesting a new value from the device. If maxAge is set to 0, the FDI Server shall read a new value from the data source. Values greater than 2^{31} -1 (0x7fff ffff) are invalid for maxAge.
Response		
readResult []	DataValue	The StatusCode, Value and timestamps for each Node Attribute that was read. The order of this list matches the order of the attributesToRead request parameter. The DataValue is defined in 5.1.9.3.3.
innerErrorInfos []	InnerErrorInfo	List of error information from calls to an underlying system. See 5.1.9.3.4. Matches the size and order of the attributesToRead request parameter. This list is empty if inner error information was not requested or if no information was encountered in the processing of the request.

5.1.6.4.3 Service results

Table 14 defines values for the Service result code. Other common StatusCodes are defined in Table 7.

Table 14 – Read Service result codes

Result code	Description
Bad_MaxAgeInvalid	The max age parameter is invalid.

5.1.6.4.4 Operation result codes

Table 15 defines values for the operation status code contained in the DataValue of each readResult element. All operational status codes with their description are in Table 8.

Table 15 – Read operation result codes

Result code
Good
Good_LocalOverride
Good_PostActionFailed
Good_DependentValueChanged
Uncertain
Uncertain_NoCommunicationLastValue
Uncertain_LastUsableValue
Uncertain_SubstituteValue
Uncertain_InitialValue
Uncertain_SensorNotAccurate
Uncertain_EngineeringUnitsExceeded
Uncertain_SubNormal
Uncertain_DominantValueChanged
Bad
Bad_UserAccessDenied
Bad_ConfigurationError
Bad_NotConnected
Bad_DeviceFailure
Bad_SensorFailure
Bad_OutOfRange
Bad_OutOfService
Bad_NodeInvalid
Bad_AttributeInvalid
Bad_IndexRangeInvalid
Bad_NotReadable

5.1.6.5 CancelRead

5.1.6.5.1 Description

Calling CancelRead indicates that the UIP has no further interest in the results of this service. Execution will be stopped when possible.

Cancel is a suggestion to the system. Due to asynchronous execution, the service may already be fully or partially completed.

5.1.6.5.2 Parameters

Table 16 defines the parameters for the service.

Table 16 – CancelRead Service parameters

Name	Type	Description
Request		
serviceId	<technology dependent>	The identifier of the service to cancel.
Response		

5.1.6.5.3 Service results

No specific Service result codes are defined for CancelRead. Successfully cancelled Read requests shall respond with Bad_RequestCancelled. Common StatusCodes are defined in Table 7.

5.1.6.6 Write

5.1.6.6.1 Description

This service is used to write values to one or more Variables. For array values, this service allows writing the entire array, writing individual elements or writing ranges of elements.

Explicit locking is required (see 5.1.7). If the Node is not locked, the request will be rejected with Bad_LockRequired.

The service response is not returned until the write operation has been executed by the system. Rollback is the responsibility of the FDI Client/UIP.

The values shall have the correct data type.

NOTE No automatic data type translation takes place (see Bad_TypeMismatch operation result code).

5.1.6.6.2 Parameters

Table 17 defines the parameters for the service.

Table 17 – Write Service parameters

Name	Type	Description
Request		
returnInnerErrorInfo	Boolean	A value of “true” requests error information from calls to an underlying system to be returned when available. “false”defines that this information shall not be returned.
variablesToWrite[]	structure	List of Variables to write to. No assumptions should be made about the order of processing this list. If the order matters, separate service requests shall be used.
node	NodeSpecifier	The identifier of the Node that contains the Attribute to write. See 5.1.9.3.2 for the definition of the NodeSpecifier type.
indexRange	NumericRange	See 5.1.9.3.6 for a detailed definition.
value	Variant	Value to write.
Response		
writeResult[]	UInt32	Status codes for operation results as defined in Table 18. The order of this list matches the order of the variablesToWrite request parameter.
innerErrorInfos []	InnerErrorInfo	List of error information from calls to an underlying system. See 5.1.9.3.4. Matches the size and order of the variablesToWrite request parameter. This list is empty if inner error information was not requested or if no information was encountered in processing of the request.

5.1.6.6.3 Service results

There are no service results other than the common codes specified in 5.1.4.6.

5.1.6.6.4 Operation result codes

Table 18 defines values for the operation status code contained in the writeResult elements. All operational status codes with their description are in Table 8.

Table 18 – Write operation result codes

Result code
Good
Good_PostActionFailed
Bad
Bad_UserAccessDenied
Bad_NodeInvalid
Bad_IndexRangeInvalid
Bad_TypeMismatch
Bad_OutOfRange
Bad_NotWritable

5.1.6.7 CancelWrite

5.1.6.7.1 Description

Calling CancelWrite indicates that the UIP has no further interest in the results of this service. Execution will be stopped when possible.

Cancel is a suggestion to the system. Due to asynchronous execution, the service may already be fully or partially completed.

5.1.6.7.2 Parameters

Table 19 defines the parameters for the service.

Table 19 – CancelWrite Service parameters

Name	Type	Description
Request		
serviceId	<technology dependent>	The identifier of the service to cancel.
Response		

5.1.6.7.3 Service results

No specific Service result codes are defined for CancelWrite. Successfully cancelled Write requests shall respond with Bad_RequestCancelled. Common StatusCodes are defined in Table 7.

5.1.6.8 Subscriptions

5.1.6.8.1 Subscription mechanism

Subscriptions allow the UIP to receive unsolicited callbacks from the FDI Client when the subscribed Node Attributes change. Subscriptions are a more efficient way to get periodic updates of data than by issuing repeated calls to the Read service, i.e., polling. The UIP creates a subscription by calling the CreateSubscription service (see 5.1.6.8.2). When the CreateSubscription service is called the UIP shall supply a callback parameter with the UIP-specific DataChangeCallback service (see 5.1.6.8.6). After receiving the subscription identifier in the CreateSubscription response the UIP shall add Node Attributes of interest to the Subscription by calling the Subscribe service.

Once the initial values have been reported, the DataChangeCallback service is only called when Node Attributes change and only the Node Attributes that have changed are reported to the callback. The UIP controls the maximum frequency at which the callback should be invoked by specifying a rate in milliseconds.

After Node Attributes have been subscribed, their values may not be immediately available and may become available at different times. As such, the initial callback may not include all of the subscribed Node Attributes. Furthermore, it is possible that the initial update for a subscribed Node Attribute will return a Bad or Uncertain StatusCode.

Furthermore, the time span between when a change happens and when the callback is invoked depends on where the information to be changed is maintained. Some values are maintained by the system, which allows immediate notification; others are located in the physical device so that communication is required to access it.

5.1.6.8.2 CreateSubscription Service

5.1.6.8.2.1 Description

This service is used to create a subscription.

5.1.6.8.2.2 Parameters

Table 20 defines the parameters for the service.

Table 20 – CreateSubscription Service parameters

Name	Type	Description
Request		
requestedUpdateRate	UInt32	The fastest rate, in milliseconds, at which the UIP requests to be called back with data changes, specified by the minimum milliseconds to elapse between updates. Regardless of the requested rate, a callback only occurs if data has changed. A rate of “0” indicates that the caller wants to be notified of changes as soon as possible. The service will return the fastest possible rate as revisedUpdateRate.
dataChangeCallback	DataChangeCallback	Callback for sending data change updates to the UIP. See 5.1.6.8.6. The DataChangeCallback is a service implemented and provided by the UIP.
Response		
revisedUpdateRate	UInt32	The actual rate that the FDI Server will use, expressed as the minimum milliseconds to elapse between updates (if data has changed since the previous update).
subscriptionId	SubscriptionId	The callee-assigned identifier for the subscription. The SubscriptionId type is technology dependent.

5.1.6.8.2.3 Service results

Table 21 defines values for the service result. Common results are defined in Table 7.

Table 21 – CreateSubscription Service result codes

Result code	Description
Bad_TooManySubscriptions	The FDI Server has reached its maximum number of subscriptions.

5.1.6.8.3 Subscribe Service

5.1.6.8.3.1 Description

This service is used to add one or more Node Attributes to an existing subscription.

Subscribing is permitted for all Node Attributes.

5.1.6.8.3.2 Parameters

Table 22 defines the parameters for the service.

Table 22 – Subscribe Service parameters

Name	Type	Description
Request		
subscriptionId	SubscriptionId	The identifier for an existing subscription that was returned by the CreateSubscription service.
returnInnerErrorInfo	Boolean	A value of "true" requests error information from calls to an underlying system to be passed in DataChangeCallbacks, when available. "false"defines that this information shall not be returned.
monitoredItemsToAdd[]	structure	The Attributes to add to the subscription.
node	NodeSpecifier	The identifier of the Node that contains the Attribute to subscribe. See 5.1.9.3.2 for the definition of the NodeSpecifier type.
attributeId	AttributeId	Numeric identifier of the Attribute to subscribe. See 5.1.9.3.3.
indexRange	NumericRange	This parameter is used to identify a single element of an array, or a single range of indexes for arrays. If a range of elements is specified, the values are returned as a composite. The first element is identified by index 0 (zero). This parameter is ignored if the specified Attribute is not an array or a structure. However, if the specified Attribute is an array or a structure, and this parameter is null, then all elements are to be included in the range. See 5.1.9.3.6 for a detailed definition.
samplingInterval	Int32	The interval that defines the fastest rate at which the Attribute should be accessed and evaluated. This interval is defined in milliseconds. The value 0 indicates that the FDI Server should use the fastest practical rate. The value -1 indicates that the default sampling interval defined by the UpdateRate of the Subscription is used. See 5.1.6.8.3.3 for further details on the sampling interval.
uIPHandle	UInt32	A handle (an identifier) provided by the UIP for the subscribed Node Attribute. This handle will be passed together with the data in the DataChangeCallback service so that the UIP can easily associate each changed value with the subscribed Node Attribute. The uiPHandle is likely an index into a table somewhere. It does not have to be unique (there could be multiple subscribed items pointing to the same table entry).
Response		
subscribeResult []	structure	List of results for the subscribed Attributes. The size and order of the list matches the size and order of the attributesToSubscribe request parameter.
statusCode	UInt32	Status code for the respective Attribute to subscribe as defined in Table 23.
monitoredItemId	UInt32	FDI Server-assigned id for the subscribed Attribute. This id is unique within the Subscription and shall be used when calling Unsubscribe. This parameter is present only if the statusCode indicates that the Attribute was successfully subscribed.
revisedSamplingInterval	Int32	The actual sampling interval that will be used. This value is based on a number of factors, including the capabilities of the underlying system.

Subscribing will succeed even if the current user is not authorized to access the Node Attribute. If this is the case, the initial callback will return the operation result "Bad_UserAccessDenied". Once this denial is cleared away (for instance, after a more powerful user identity is provided) the UIP will receive data changes for this Node Attribute.

It is possible to subscribe to any Attribute – not just the Value. While it may not make sense for all Attributes, monitoring of some Attributes provides additional possibilities. Some examples include:

- monitoring the LockedStatus to determine when the lock by some other client is removed;
- monitoring the CurrentLabel of Enumerations to receive a displayable name rather than a numeric value.

5.1.6.8.3.3 Sampling interval

Each subscribed item is assigned a sampling interval that is either inherited from the updateRate of the Subscription or that is defined specifically to override that rate. The sampling interval indicates the fastest rate at which the value should be sampled in the device for data changes.

The assigned sampling interval defines a "best effort" cyclic rate that is used to sample the item from its source. "Best effort" in this context means that the system does its best to sample at this rate. Sampling at rates faster than this rate is acceptable, but not necessary to meet the needs of the UIP. How the system deals with the sampling rate and how often it actually polls its data source internally is a system implementation detail. However, the time between values returned to the UIP shall be greater than or equal to the sampling interval.

The FDI Client may also specify 0 for the sampling interval, which indicates that the system should use the fastest practical rate. It is expected that systems will support only a limited set of sampling intervals to optimize their operation. If the exact interval requested by the UIP is not supported, then the most appropriate interval as determined by the system will be assigned and returned to the UIP.

Data may be collected based on a sampling model or generated based on an exception-based model. The fastest supported sampling interval may be equal to 0, which indicates that the data item is exception-based rather than being sampled at some period. When it is exception-based the underlying system does not require sampling.

In many cases, the system has no knowledge of the data update logic. In this case, even though the system samples at the negotiated rate, the data might be updated by the underlying system at a much slower rate. In this case, changes can only be detected at this slower rate.

UIPs should also be aware that the sampling by the system and the update cycle of the device are usually not synchronized, which may lead to additional delays.

5.1.6.8.3.4 Service results

There are no service results other than the common codes specified in 5.1.4.6.

5.1.6.8.3.5 Operation result codes

Table 23 defines values for the operation statusCode contained in the results. All operational status codes with their description are in Table 8.

Table 23 – Subscribe operation result codes

Result code
Good
Bad
Bad_NodeInvalid
Bad_AttributeInvalid
Bad_NotReadable
Bad_UserAccessDenied

5.1.6.8.4 Unsubscribe Service

5.1.6.8.4.1 Description

This service is used to unsubscribe to one or more Node Attributes.

5.1.6.8.4.2 Parameters

Table 24 defines the parameters for the service.

Table 24 – Unsubscribe Service Parameters

Name	Type	Description
Request		
subscriptionId	SubscriptionId	The identifier for an existing subscription that was returned by the CreateSubscription service.
monitoredItemIds[]	UInt32	Identifiers for subscribed Attributes that were returned by the Subscribe service.
Response		
unsubscribeResult[]	UInt32	Status codes for operation results as defined in Table 25. The order of this list matches the order of the monitoredItemIds request parameter.

5.1.6.8.4.3 Service results

There are no service results other than the common codes specified in 5.1.4.6.

5.1.6.8.4.4 Operation result codes

Table 25 defines values for the operation status code contained in unsubscribeResult. All operational status codes with their description are in Table 8.

Table 25 – Unsubscribe operation result codes

Result code
Good
Bad
Bad_UIPHandleInvalid

5.1.6.8.5 DeleteSubscription Service

5.1.6.8.5.1 Description

This service is used to delete a subscription. Due to the asynchronous nature of the callbacks, the UIP may receive additional callbacks for a subscription after the subscription is deleted

5.1.6.8.5.2 Parameters

Table 26 defines the parameters for the service.

Table 26 – DeleteSubscription Service parameters

Name	Type	Description
Request		
subscriptionId	SubscriptionId	The identifier for an existing subscription that was returned by the CreateSubscription service.
Response		

5.1.6.8.5.3 Service results

There are no service results other than the common codes specified in 5.1.4.6.

5.1.6.8.6 DataChangeCallback Service

5.1.6.8.6.1 Description

This service is used for sending data change updates to the UIP. This service is implemented and provided by the UIP when calling the CreateSubscription service.

Due to the asynchronous nature of the callbacks, the UIP may receive additional callbacks for a subscription after the subscription is deleted.

5.1.6.8.6.2 Parameters

Table 27 defines the parameters for the service.

Table 27 – DataChangeCallback Service parameters

Name	Type	Description
Request		
subscriptionId	SubscriptionId	Identifier that was returned by the CreateSubscription service
dataChangeData[]	structure	Data that has changed. No specific order of array elements is ensured.
uIPHandle	UInt32	The handle provided by the UIP in the Subscribe service
value	DataValue	The StatusCode, Value and timestamps of the subscribed Node Attribute. The DataValue is defined in 5.1.9.3.3.
innerErrorInfos []	InnerErrorInfo	List of error information from calls to an underlying system. See 5.1.9.3.4. Matches the size and order of the dataChangeData parameter. This list is empty if inner error information was not requested or if no information was encountered in the processing of the request.
Response		

5.1.6.8.6.3 Operation result codes

Table 28 defines values for the operation status code contained in the DataChangeData structure of each values element. In addition, all Read operation status codes apply also (see Table 15). All operational status codes with their description are in Table 8.

Table 28 – DataChangeCallback result codes

Result code
Bad_WaitingForInitialData

5.1.7 Locking Services

5.1.7.1 Overview

Locking is the means to avoid concurrent modifications to a Device or a Block and its Parameters. UIPs shall use the locking services before making changes (for example, write operations and Direct Access Services).

The lock always applies to both the online and the offline version.

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

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

While locked, requests from other FDI Clients to write to Parameters, or to perform Direct Access will be rejected.

The lock is removed when ExitLock is called.

5.1.7.2 InitLock service

5.1.7.2.1 Description

InitLock reserves the specified Device or Block. During a lock other FDI Clients will not be able to write to the Parameters of this element. A lock for an element that is already locked by another FDI Client will be rejected. A UIP can subscribe to the LockedStatus Attribute in order to be informed when the lock is removed by the other Client.

FDI Clients shall allow “nested” InitLock calls from the same UIP. The FDI Client expects the same number of ExitLock calls before it actually removes the lock.

FDI Clients are responsible for preventing simultaneous locks to a Device or Block by independent components that this Client hosts. Such components include the Client itself, independent UIPs or the UID Interpreter.

5.1.7.2.2 Parameters

Table 29 defines the parameters for the service.

Table 29 – InitLock Service parameters

Name	Type	Description
Request		
node	NodeSpecifier	The identifier of the Node (representing a device or block) to be locked. See 5.1.9.3.2 for the definition of the NodeSpecifier type.
context	String	Used to provide context information about the current activity going on in the UIP. This will be used to enhance the Audit Trail.
Response		

5.1.7.2.3 Service results

Table 30 defines values for the Service result code. Other common StatusCodes are defined in Table 7.

Table 30 – InitLock Service result codes

Result code	Description
Bad_NotSupported	The Node does not support locking.
Bad_AlreadyLocked	The Node is already locked by another FDI Client or another independent component within the FDI Client.

5.1.7.3 ExitLock service

5.1.7.3.1 Description

ExitLock removes the lock.

5.1.7.3.2 Parameters

Table 31 defines the parameters for the service.

Table 31 – ExitLock Service parameters

Name	Type	Description
Request		
node	NodeSpecifi er	The identifier of the Node (representing a device or block) to be unlocked. See 5.1.9.3.2 for the definition of the NodeSpecifier type.
Response		

5.1.7.3.3 Service results

Table 32 defines values for the Service result code. Other common StatusCodes are defined in Table 7.

Table 32 – ExitLock Service result codes

Result code	Description
Bad_ InvalidState	The Node is not locked.

5.1.8 Direct Access Services

5.1.8.1 Overview

Direct Access Services provide direct communication with a Device. This can be used for operations that cannot or at least not easily be performed through the Device Model Services. Use cases include the transmission of large data buckets from or to the Device, for example, historical data or firmware. The Direct Access Services should not influence the structure and the data integrity of the Device Model. Support of Direct Access Services is mandatory for FDI Hosts. However, Host Systems shall provide means to disable/enable Direct Access Services on demand. Commissioning engineers or plant operators can then disallow the use of Direct Access in specific scenarios. This can be temporary or even permanent and might apply to the complete or specific parts of the plant. Therefore, UIPs shall not depend on the availability of Direct Access for the initial setup of a Device (e.g. needed for commissioning).

The following behaviour applies when using Direct Access.

- Only one FDI Client/UIP can use DirectAccess at a given time. While in Direct Access mode, other Clients will not be able to access this Device (not even for Reading).
- The Device shall have been locked prior to entering this mode.
- If Variable Attributes may have changed due to DirectAccess, the UIP shall set the InvalidateCache in the EndDirectAccess argument to “true”.

The Direct Access Services include:

- InitDirectAccess
- Transfer
- ExitDirectAccess

Guideline:

These services are not to be used as alternative to Information Model access. Instead, DirectAccess is used for the transfer of data that are not reflected in the Information Model.

5.1.8.2 InitDirectAccess

5.1.8.2.1 Description

This service initializes the Device for the use of DirectAccess.

5.1.8.2.2 Parameters

Table 33 defines the parameters for the service.

Table 33 – InitDirectAccess Service parameters

Name	Type	Description
Request		
context	String	Used to provide context information about the current activity going on in the UIP. This will be used to enhance the Audit Trail.
Response		

5.1.8.2.3 Service results

Table 34 defines values for the Service result code. Other common StatusCodes are defined in Table 7.

Table 34 – InitDirectAccess Service result codes

Result code	Description
Bad_NotSupported	DirectAccess is (currently) not supported.
Bad_LockRequired	The Node has not been locked.
Bad_InvalidState	DirectAccess is already initialized.

5.1.8.3 ExitDirectAccess

5.1.8.3.1 Description

This service ends the use of DirectAccess.

5.1.8.3.2 Parameters

Table 35 defines the parameters for the service.

Table 35 – ExitDirectAccess Service parameters

Name	Type	Description
Request		
invalidateCache	Boolean	If “true”, any cached values for Device Parameters will be invalidated. This means that these Parameters will be re-read from the Device the next time they are used again.
Response		

5.1.8.3.3 Service results

Table 36 defines values for the Service result code. Other common StatusCodes are defined in Table 7.

Table 36 – ExitDirectAccess Service result codes

Result code	Description
Bad_InvalidState	The device is not in DirectAccess mode.

5.1.8.4 Transfer

5.1.8.4.1 Description

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

Direct Access by its nature does not allow automatic generation of Audit Trail information that complies with the various regulations. Therefore, UIPs shall invoke the LogAuditTrailMessage Service (see 5.2.2.5) to provide explicit information about what is being transferred.

For other service calls that affect the Device indirectly via the Device Model (Write), the service parameters provide sufficient information.

5.1.8.4.2 Parameters

Table 37 defines the parameters for the service.

Table 37 – Transfer Service parameters

Name	Type	Description
Request		
sendData	String	XML document based on the TransferSendDataT as specified in the communiction profile-specific XML schema.
Response		
receiveData	String	XML document based on the TransferResultDataT as specified in the communiction profile-specific XML schema.

5.1.8.4.3 Service results

Table 38 defines values for the Service result code. Other common StatusCodes are defined in Table 7.

Table 38 – Transfer Service result codes

Result code	Description
Bad_InvalidState	The device is not in DirectAccess mode.

5.1.9 Data types

5.1.9.1 General

Subclause 5.1.9 specifies the data types used for Service parameters, Variable values, and the values of other Node Attributes. They may occur either scalar or as an array.

5.1.9.2 Base data types

Table 39 lists base data types, i.e., types that are typically supported native by programming languages.

Table 39 – Base data types

DataType	Description
Boolean	Defines a value that is either “true” or “false”.
String	Represents text as a series of Unicode characters. The actual string representation depends on the technology mapping. See IEC 62769-6.
ByteString	A value that is a sequence of Byte values preceded by a 32-Bit Length.
UtcTime	A DateTime used to define Coordinated Universal Time (UTC) values. All time values are UTC values. FDI Clients shall provide any conversions between UTC and local time. UtcTime is a 64-bit signed integer that represents the number of 100 nanosecond intervals since January 1, 1601. It corresponds to the Windows FILETIME.
Int8	A signed integer between -128 and 127 inclusive.
Int16	A signed integer between -32 768 and 32 767 inclusive.
Int32	A signed integer between -2 147 483 648 and 2 147 483 647 inclusive.
Int64	A signed integer between -9 223 372 036 854 775 808 and 9 223 372 036 854 775 807 inclusive.
Byte	A value in the range of 0 to 255.
UInt16	An unsigned integer between 0 and 65 535 inclusive.
UInt32	An unsigned integer between 0 and 4 294 967 295 inclusive.
UInt64	An unsigned integer between 0 and 18 446 744 073 709 551 615 inclusive.
Float	Defines a value that shall be according to the IEEE 754 single precision data type definition.
Double	Defines a value that shall be according to the IEEE 754 double precision data type definition.
Duration	Same representation as Double.

5.1.9.3 Special types

5.1.9.3.1 AttributeIds

AttributeIds are represented as UInt32. Table 40 lists the Attributes and their identifiers.

Table 40 – Identifiers assigned to Attributes

Attribute	Identifier
Name	10
Label	11
Description	12
	…
LockedStatus	30
	…
Value	100
DataType	101
ValueRank	102
ArrayDimensions	103
AccessRights	105
UserAccessRights	106
ScalingFactor	107
	…
EURange	111
EngineeringUnits	112
	…
EnumValues	120
CurrentLabel	121
OptionNames	122

5.1.9.3.2 NodeSpecifier

Each Node in the Device Model (see 5.1.2) is uniquely addressable with its path name qualified by an online/offline specifier. When online is specified, the service shall operate in the online version of the Device Model. When offline is specified, it shall operate in the offline model.

The path name follows the hierarchy of the device model as illustrated in 5.1.2, It is a concatenation of the individual names, separated by slash ('/') characters. See 5.1.2 for example path names. All parameters are identified via "/ParameterSet/<ParamName>" , all images via "/ImageSet/<ImageName>", and so on.

Subclauses 5.1.3.4.3 and 5.1.3.4.5 specify the representation of parameters that hold record values or arrays of record values. A pathname for record elements is built by extending the path to the Parameter. See examples in the referenced subclauses. The components of this parameter are defined in Table 42.

Table 41 – NodeSpecifier

Name	Type	Description
NodeSpecifier	structure	Specifies a Node in the device model.
nodePath	String	The path description that enables finding a Node in the Information Model.
useOnline	Boolean	If "true" the path addresses a Node in the online model; with "false" a Node in the offline model is referenced.

An empty path name identifies the Root.

5.1.9.3.3 DataValue

This type describes the value of a Node Attribute in the response of a Read and in the DataChangeCallback. The components of this parameter are defined in Table 42.

Table 42 – DataValue

Name	Type	Description
DataValue	structure	The value and associated information.
value	Variant	The Node Attribute value. For the definition of Variant see 5.1.9.4.
statusCode	UInt32	The StatusCode that defines the ability to access/provide the value. The StatusCode type is defined in 5.1.4.5.
sourceTimestamp	UtcTime	The source timestamp for the value.
serverTimestamp	UtcTime	The server timestamp for the value.

The statusCode is used to indicate the conditions under which a Node Attribute value was generated, and thereby can be used as an indicator of the usability of the value.

It is required to check the StatusCode (as a minimum the Severity) of all results before accessing and using the value.

The sourceTimestamp reflects the timestamp that was applied by the data source. The sourceTimestamp is only returned with the Value Attribute of Variable Nodes. For all other Attributes the returned sourceTimestamp is set to null.

In the case of a bad or uncertain status sourceTimestamp is used to reflect the time that the source recognized the non-good status or the time the FDI Server last tried to recover from the bad or uncertain status.

The serverTimestamp is used to reflect the time that the FDI Server received a Variable value or knew it to be accurate. In the case of a bad or uncertain status, serverTimestamp is used to reflect the time that the FDI Server received the status or that the FDI Server last tried to recover from the bad or uncertain status.

The serverTimestamp is updated each time a new value is received.

5.1.9.3.4 InnerErrorInfo

The Services described in 5.1 return StatusCodes on the service level and on the operation level. These StatusCodes are protocol-independent and device-independent. In cases where a StatusCode results from a call to the underlying system (e.g., communication system, device) the status of the underlying system can be reported via InnerErrorInfo.

The components of this parameter are defined in Table 43.

Table 43 – InnerErrorInfo

Name	Type	Description
InnerErrorInfo	structure	Communication- or Device-specific information.
symbolicId	String	This string shall be used to identify the result of some internal operation. The maximum length of this string is 32 characters. Systems wishing to return a numeric return code should convert the return code into a string and use this string as symbolicId (e.g., "0xC0040007" or "-4").
errorText	LocalizedText	A textual representation of the symbolic id. The maximum length of this text string is 256 characters

5.1.9.3.5 LocalizedText

This data type defines a structure containing a String in a locale-specific translation specified in the identifier for the locale. Its elements are defined in Table 44.

Table 44 – LocalizedText Definition

Name	Type	Description
LocalizedText	structure	—
text	String	The localized text.
locale	String	The identifier for the locale (e.g. “en-US”).

The element locale shall be a string composed of a language component and a country/region component according to RFC 3066. The <country/region> component is always preceded by a hyphen. The format of the LocaleId string is shown below:

<language>[-<country/region>], where
<language> shall be a two-letter code for a language according to ISO 639,
<country/region> shall be a two-letter code for the country/region according to ISO 3166.

The rules for constructing LocaleIds shall be according to RFC 3066 and shall be restricted as follows:

- This specification permits only zero or one <country/region> component to follow the <language> component.
- This specification also permits the “-CHS” and “-CHT” three-letter <country/region> codes for “Simplified” and “Traditional” Chinese locales.
- This specification also allows the use of other <country/region> codes as deemed necessary by the FDI Client or the FDI Server.

Table 45 shows examples of locale ids.

Table 45 – LocaleId Examples

Locale	OPC UA LocaleId
English	en
English (US)	en-US
German	de
German (Germany)	de-DE
German (Austrian)	de-AT

An empty or NULL string indicates that the locale is unknown.

5.1.9.3.6 Numeric Range

Numeric Range is represented as a String. The syntax for the String contains one of the following two constructs. The first construct is the String representation of an individual integer. For example, “6” is valid, but “6,0” and “3,2” are not. The minimum and maximum values that can be expressed are defined by the use of this parameter and not by this parameter type definition. The second construct is a range represented by two integers separated by the colon (“:”) character. The first integer shall always have a lower value than the second. For example, “5:7” is valid, while “7:5” and “5:5” are not. No other characters, including white-space characters, are permitted.

All indexes start with 0. The maximum index is one less than the length of the array.

When reading with a numeric range outside the bounds of the array, the FDI Server shall return a partial result if some elements exist within the range. The FDI Server shall return a Bad_OutOfRange if no elements exist within the range.

When writing a value the numeric range shall be within the array.

A numeric range can also be used to specify substrings for ByteString and String values.

5.1.9.3.7 Range

This structure defines the range structure needed for the EURange Attribute. It is defined in Table 46.

Table 46 – Range Data Type Structure

Name	Type	Description
Range	structure	“low” and “high” can contain any data type that is appropriate for the data type of the Variable Value Attribute.
low	Variant	Lowest value in the range.
high	Variant	Highest value in the range.

5.1.9.3.8 EUInformation

This structure contains information about the EngineeringUnits. Its elements are defined in Table 47.

Table 47 – EUInformation Data Type Structure

Name	Type	Description
EUInformation	structure	—
unitId	UInt32	Identifier for programmatic evaluation. 0 is used if a unitId is not available.
displayName	LocalizedText	The displayName of the engineering unit is typically the abbreviation of the engineering unit, e.g. ”h” for hour or ”m/s” for meter per second.
description	LocalizedText	Contains the full name of the engineering unit such as hour or meter per second. An empty text field indicates that no description is available.

5.1.9.3.9 EnumValueType

This Structured DataType is used to represent a human-readable representation of an Enumeration. Its elements are described in Table 48. When this type is used in an array representing human-readable representations of an enumeration, each value of an IntegerRepresentation will be unique in that array.

Table 48 – EnumValueType Definition

Name	Type	Description
EnumValueType	structure	—
value	Int64	The Integer representation of an Enumeration.
displayName	LocalizedText	A human-readable representation of the integer representation of the Enumeration.
description	LocalizedText	A localized description of the enumeration value. This field can contain an empty String if no description is available.

5.1.9.4 Variant

A Variant is a union of all data types. Variants can also contain arrays of any of these types.

Variants can be empty. An empty Variant is described as having a Null value. A Null value in a Variant is not the same as a Null String.

Variants can contain arrays of Variants but they cannot directly contain another Variant.

5.2 Hosting Services

5.2.1 General

The Hosting Services are provided by the FDI Client for use by the UIP. The Hosting Services include services related to the FDI Client environment allowing the UIP to acquire information about the environment. The Hosting Services also include services to launch other UIPs as well as showing feedback.

5.2.2 Services

5.2.2.1 General

Programmatic access to the services may be synchronous or asynchronous.

5.2.2.2 GetClientTechnology Version

5.2.2.2.1 Description

This service returns the technology version of the FDI Client that hosts the UIP.

5.2.2.2.2 Parameters

Table 49 defines the parameters for the service.

Table 49 – GetClientTechnologyVersion Service parameters

Name	Type	Description
Request		
Response		
version	String	FDI Technology Version that is supported by the FDI Client. The format of the value is xx.yy.zz as defined in IEC 62769-4.

5.2.2.2.3 Service results

This service always succeeds.

5.2.2.3 OpenUserInterface Service

5.2.2.3.1 Description

This service is used by a UIP to request the FDI Client to open another UIP. The UIP is opened in either a modal or a non-modal window as follows:

- UIPs are always invoked in a modal window, if the calling UIP runs in a modal window or if their style is dialog.
- UIPs are invoked non-modal if their style = window and the calling UIP runs in non-modal mode also.

If the UIP is already opened, this service shall bring the respective window into the foreground.

5.2.2.3.2 Parameters

Table 50 defines the parameters for the service.

Table 50 – OpenUserInterface Service parameters

Name	Type	Description
Request		
uipID	String	Identification of the UIP to be opened. This string is a UUID that defines a Node in the Information Model. The value of this UUID is defined in the FDI Package corresponding to the UIP to be opened (see IEC 62769-4).
Response		
Errors		

5.2.2.4 CloseUserInterface Service

5.2.2.4.1 Description

This service is used by a UIP to close itself. The UIP shall finish all pending or running functions that are still open.

UIPs call this Service only on user request (Ok, Close or Cancel).

5.2.2.4.2 Parameters

There are no specific parameters for the service.

5.2.2.5 LogAuditTrailMessage Service

5.2.2.5.1 Description

This service is used by a UIP to log an audit trail message.

5.2.2.5.2 Parameters

Table 51 defines the parameters for the service.

Table 51 – LogAuditTrailMessage Service parameters

Name	Type	Description
Request		
message	String	The message to be logged in the audit trail.
Response		None.

5.2.2.6 SaveUserSettings Service

5.2.2.6.1 Description

This service is used by a UIP to instruct the FDI Client to save the supplied user settings. The settings are stored per UIP type in a persistent store under the control of the FDI Client. All previously saved user settings of this UIP type are replaced. The FDI Client shall not alter the user settings requested of the storage.

Typically, user settings include layout information or other user preferences. They are not designed to be used for instance-specific data.

There is no support for partial modifications of the user settings. A UIP has to load all settings, update them and save the complete set.

The structure and content of the user setting strings is UIP type specific. For example, a UIP could use name-value pairs. Versioning issues of this data are under the responsibility of the UIP. Different UIPs of the same type can overwrite each other’s settings.

Data structure is completely under control of the UIP. Therefore also versioning issues with respect to different UIP versions are under the responsibility of the UIP.

5.2.2.6.2 Parameters

Table 52 defines the parameters for the service.

Table 52 – SaveUserSettings Service parameters

Name	Type	Description
Request		
userSetting[]	String	List of user settings. The content of the strings is UIP-type specific.
Response		

5.2.2.7 LoadUserSettings Service

5.2.2.7.1 Description

This service is used by a UIP to instruct the FDI Client to retrieve the previously saved user settings.

5.2.2.7.2 Parameters

Table 53 defines the parameters for the service.

Table 53 – LoadUserSettings Service parameters

Name	Type	Description
Request		
Response		
userSetting[]	String	List of user settings. The content of the strings is UIP-specific.

5.2.2.8 Trace Service

5.2.2.8.1 Description

This service shall be used by the UIP to provide the FDI Client with information about internal events in the UIP. The trace messages are typically used for trouble shooting. The UIP shall call this service according to the trace level settings specified in the UIP Service SetTraceLevel (see 6.1.1.4).

5.2.2.8.2 Parameters

Table 54 defines the parameters for the service.

Table 54 – Trace Service parameters

Name	Type	Description
Request		
eventType	TraceLevel	Severity of the trace message. One of the values that specifies the severity of the trace data (see 6.1.2).
classification	String	UIP-specific classification of the trace message.
message	String	Trace message. The trace message language shall be English. Embedded text might be localized.
Response		

5.2.2.9 ShowMessageBox Service

5.2.2.9.1 Description

This service opens a message box and waits until the user presses one of the given buttons. An open message box shall block any activity for the related device instance. It is recommended that simultaneous interactions with other Device instances are not affected.

5.2.2.9.2 Parameters

Table 55 defines the parameters for the service.

Table 55 – ShowMessageBox Service parameters

Name	Type	Description
Request		
acknStyle	AcknStyle	See Table 63.
message	String	Message to be shown to the user.
caption	String	Title bar caption to display.
buttonSet	ButtonSet	See Table 62.
defaultResult	Integer	Id of the default button; see Table 61.
Response		
buttonSelected	Integer	Id of the selected button; see Table 61.

5.2.2.10 ShowProgressBar Service

5.2.2.10.1 Description

This service opens a progress bar. The progress bar remains on the user interface after the service returns. The information shown can be updated with the UpdateShowProgressBar service (see 5.2.2.11). The progress bar can be closed with the EndShowProgressBar service (see 5.2.2.12).

Only one progress bar per UIP can be shown at a time.

5.2.2.10.2 Parameters

Table 56 defines the parameters for the service.

Table 56 – ShowProgressBar Service parameters

Name	Type	Description
Request		
message	String	Message to be shown to the user.
callback	CancelCallback	Service called by the Client to inform the UIP that the user has cancelled the operation. See 0. The CancelCallback service is implemented and provided by the UIP.
Response		

5.2.2.11 UpdateShowProgressBar Service

5.2.2.11.1 Description

This service updates an already open progress bar.

5.2.2.11.2 Parameters

Table 57 defines the parameters for the service.

NOTE A call of this service is rejected without a preceeding ShowProgressBar service call.

Table 57 – UpdateShowProgressBar Service parameters

Name	Type	Description
Request		
message	String	Updated message to be shown to the user
percentage	Integer	Updated percentage of progress to be shown to the user
Response		

5.2.2.12 EndShowProgressBar Service

5.2.2.12.1 Description

This service closes an open progress bar. The service will wait until the user has pressed a button or will immediately return with the result if the user has previously pressed the button.

5.2.2.12.2 Parameters

Table 58 defines the parameters for the service.

Table 58 – EndShowProgressBar Service parameters

Name	Type	Description
Request		
message	String	Updated message to be shown to the user
percentage	Integer	Updated percentage of progress to be shown to the user
Response		

5.2.2.13 CancelCallback Service

5.2.2.13.1 Description

With this service the Client informs the UIP that the User requested to cancel the operation. This service is implemented and provided by the UIP when calling the ShowProgressBar service. The UIP is responsible for closing the ProgressBar.

Due to the asynchronous nature of callbacks, the CancelCallback might be called even after the UIP has called the EndShowProgressBar service.

5.2.2.13.2 Parameters

There are no specific parameters for the service.

5.2.2.14 StandardUIActionItemsChangeCallback Service

5.2.2.14.1 Description

This service is used by the UIP to notify the FDI Client about the change in the standard UI action items' state (enabled/disabled). See 6.1.2.2 for standard UI action items.

5.2.2.14.2 Parameters

Table 59 defines the parameters for the service.

Table 59 – StandardUIActionItemsChange Service parameters

Name	Type	Description
Request		
Response		
StandardUIActionItems[]	StandardUIActionItem	Updated list of Standard Actions provided by the UIP.

5.2.2.15 SpecificUIActionItemsChangeCallback Service

5.2.2.15.1 Description

This service is used by the UIP to notify FDI Client about the change in the UI action items that are specific to this UIP. The UIP shall use this callback whenever there is addition or removal of action items or whenever there is a change in the state of an action item (i.e. enabled/disabled). See 6.1.2.4 for specific UI actions.

5.2.2.15.2 Parameters

Table 60 defines the parameters for the service.

Table 60 – SpecificUIActionItemsChange Service parameters

Name	Type	Description
Request		
Response		
SpecificUIActionItems[]	SpecificUIActionItem	Updated list of UI action Items specific to the UIP.

5.2.3 Parameter Type Definitions

5.2.3.1 DefaultResult Definition

The components of this parameter are defined in Table 61.

Table 61 – DefaultResult definition

Name	Type	Description
DefaultResult	Integer	Definition of a button on a message box. Used to specify the default button and the button selected by the user. This value is an enumeration with one of the following values: BUTTONNONE_0 BUTTONOK_1 BUTTONCANCEL_2 BUTTONYES_3 BUTTONNO_4

5.2.3.2 ButtonSet

The components of this parameter are defined in Table 62.

Table 62 – ButtonSet definition

Name	Type	Description
ButtonSet	Integer	Definition of the buttons on a message box or progress bar shown to the user. This value is an enumeration with one of the following values: BUTTONSETOK_0 Only ok button is shown BUTTONSETOKCANCEL_1 Ok and cancel buttons are shown BUTTONSETYESNOCANCEL_2 Yes, no, and cancel buttons are shown BUTTONSETYESNO_3 Yes and no buttons are shown

5.2.3.3 AcknStyle

The components of this parameter are defined in Table 63.

Table 63 – AcknStyle definition

Name	Type	Description
AcknStyle	Integer	The style of the message box or progress bar shown to the user. For example, this parameter may influence the icon of the message box. This value is an enumeration with one of the following values: ACKNSTYLEINFO_0 Information style ACKNSTYLEWARNING_1 Warning style ACKNSTYLERROR_2 Error style

6 UIP

6.1 UIP Services

6.1.1 Services

6.1.1.1 Activate Service

6.1.1.1.1 Description

This service is used by the FDI Client to initialize a UIP (see 6.1.2). The FDI Client shall call this service after the creation of an instance of a UIP. The following rules define the window type:

- A modal window is used, if the UIP style = dialog, or the invoking parent (UID or UIP) is modal.
- A non-modal window is used if the UIP style = window and the invoking parent runs in non-modal mode also.

6.1.1.1.2 Parameters

Table 64 defines the parameters for the service.

Table 64 – Activate Service parameters

Name	Type	Description
Request		
hostingInterface	Interface	FDI Client hosting the interface to be used by the UIP (see 5.2).
deviceAccessInterface	Interface	FDI Client device access interface to be used by the UIP (see 5.1).
context	UInteger	Specifies in which context the UIP is activated: 0_ONLINE 1_OFFLINE The Client shall derive the context from the FunctionalGroup where this UIP has been retrieved from, i.e., whether the FunctionalGroup is part of the Offline device representation or the online device representation. If a UIP is invoked from another UIP with the OpenUserInterface or OpenModalUserInterface service, the context is inherited.
localeSetting	<technology dependent>	The locale description consists of at least a language identifier, display information and country. The format of the property value is technology dependent.
Response		

The localeSetting shall not depend on the operating system settings. The operating system settings can be used by default but shall be alterable by the user.

The language identifier shall indicate the language for localized textual information.

The display information shall be used to control colors (e.g. background color, text color for different states) and the format of specific elements (e.g. for numbers and dates).

Parameter country shall be used to apply country-specific regulations such as allowed engineering units. To do this, the UIP will modify the EngineeringUnit Attribute of correlated Variable Nodes to match the specified country. The FDI Server is then responsible to reformat the values accordingly.

6.1.1.2 Deactivate Service

6.1.1.2.1 Description

This service is used by the FDI Client to deactivate a UIP. The UIP shall release all references to other components and finish all pending or running functions including UIPs that it opened that are still open.

The UIP may indicate that it is not ready to be deactivated. This shall be used if cancelling pending or running functions would have severe side-effects.

6.1.1.2.2 Parameters

Table 65 defines the parameters for the service.

Table 65 – Deactivate Service parameters

Name	Type	Description
Request		
Response		

deactivateCancelled	Boolean	Indication whether the UIP refused the Deactivate request.
		deactivateCancelled = “true”denotes that the UIP refused the Deactivate request.
		deactivateCancelled = “false”denotes that the UIP accepted the Deactivate request.

6.1.1.3 SetSystemLabel Service

6.1.1.3.1 Description

This service is used by the FDI Client to specify a human identifier of the UIP instance in the context of the FDI Client.

The label shall be used for user interface elements displayed and managed by the UIP (e.g. a message box) and will assure that the user can uniquely identify to which Device the user interface element is directed. The UIP can extend this label with specific information when appropriate.

The FDI Client shall call this service before the Activate service (see 6.1.1.1). If it has not been called, the UIP shall use the text portion of the Label Attribute of the Device by default (see 5.1.3.2)

6.1.1.3.2 Parameters

Table 66 defines the parameters for the service.

Table 66 – SetSystemLabel Service parameters

Name	**Type**	**Description**
Request		
systemLabel	String	Human readable label that identifies the UIP within the FDI Client.
Response		None.

6.1.1.4 SetTraceLevel Service

6.1.1.4.1 Description

This service is used by the FDI Client to notify the UIP of the types of Trace messages that should be logged via the Trace service (see 5.2.2.8). Multiple types can be set (for instance: Critical, Error, and Warning).

6.1.1.4.2 Parameters

Table 67 defines the parameters for the service.

Table 67 – SetTraceLevel Service parameters

Name	**Type**	**Description**
Request		
traceLevel	TraceLevel	Trace level that shall control the type of information (see Table 72).
Response		

6.1.1.5 GetStandardUIActionItems Service

6.1.1.5.1 Description

This service is used by the FDI Client to get the available standard UI actions in the UIP (for instance: Close, Apply, Help). The FDI Client provides consistent representation of these standard UI actions, in the FDI Client UI area.

6.1.1.5.2 Parameters

Table 68 defines the parameters for the service.

Table 68 – GetStandardUIActionItems Service parameters

Name	Type	Description
Request		
Response		
StandardUIActionItems[]	StandardUIActionItem	List of Standard Actions provided by the UIP.

6.1.1.6 GetSpecificUIActionItems Service

6.1.1.6.1 Description

This service is used by the FDI Client to get the available UI actions that are specific for the UIP. The FDI Client provides consistent representation of these UI actions, in the FDI Client UI area.

6.1.1.6.2 Parameters

Table 69 defines the parameters for the service.

Table 69 – GetSpecificUIActionItems Service parameters

Name	Type	Description
Request		
Response		
SpecificUIActionItems[]	SpecificUIActionItem	List of UI actions specific for the UIP.

6.1.1.7 InvokeStandardUIAction Service

6.1.1.7.1 Description

This service is used by the FDI Client to notify the UIP to perform a standard action (for instance: Close, Apply, Help). The FDI Client passes the type of standard action to be performed in this service.

6.1.1.7.2 Parameters

Table 70 defines the parameters for the service.

Table 70 – InvokeStandardUIAction Service parameters

Name	Type	Description
Request		
actionId	StandardUIAction	One of the values defined in 6.1.2.2.
Response		

6.1.1.8 InvokeSpecificUIAction Service

6.1.1.8.1 Description

This service is used by the FDI Client to notify the UIP to perform a UI action that is specific to this UIP. The FDI Client passes the identifier of the action to be performed in this service.

6.1.1.8.2 Parameters

Table 71 defines the parameters for the service.

Table 71 – InvokeSpecificUIAction Service parameters

Name	Type	Description
Request		
actionId	UInteger	This is the identifier of the specific UI action to be performed.
Response		

6.1.2 Parameter type definitions

6.1.2.1 TraceLevel

TraceLevel is defined in Table 72.

Table 72 – TraceLevel definition

Name	Type	Description
TraceLevel	UInteger	Severity level that controls the type of information that is passed to the Trace service (see 5.2.2.8). This value is a bit enumeration with one of the following values: NONE_0 Completely switch off tracing CRITICAL_1 Fatal error or application crash ERROR_2 Recoverable error WARNING_4 Noncritical error INFO_8 Informational message VERBOSE_16 Debugging trace

6.1.2.2 StandardUIAction

Standard actions are defined to allow consistency in appearance and processing. They will be requested by the hosting Client. These actions are not expected to be mapped 1:1 to user interface buttons. Rather, the Client will display OK, CANCEL, and APPLY.

Clicking OK, CANCEL or APPLY by the User causes the following actions to be called.

- OK will cause a call to the Apply action followed by a call to the Close action.
- APPLY will cause a call to the Apply action.
- CANCEL will cause a call to the Close action. If changes are still outstanding, the UIP shall open a dialog asking the user to confirm.

The StandardUIAction enumeration is defined in Table 73.

Table 73 – StandardUIAction definition

Name	Type	Description
StandardUIAction	UInteger	Identifier for the standard UI action item with one of the following values: APPLY_0 Apply Standard UI Action • With this action the Client requests that parameter changes in the user interface will be applied to the source. CLOSE_1 Close Standard UI Action • With this action the Client requests that the UIP be closed. If changes to the parameters have not been applied, the UIP shall open a dialog asking the user to confirm. HELP_2 Help Standard UI Action • Requests to display online help describing the UIP functionality (a help document). It is no substitute for context-sensitive help which will typically be provided via tooltips.

6.1.2.3 StandardUIActionItem

The components of this parameter are defined in Table 74.

Table 74 – StandardUIActionItem definition

Name	Type	Description
StandardUIActionItem	Structure	
actionId	StandardUIAction	Identifier for the standard UI action item.
enabled	Boolean	Indicates the state of the action item. “true” indicates the item is enabled. “false” indicates the item is disabled.

6.1.2.4 SpecificUIActionItem

The components of this parameter are defined in Table 75.

Table 75 – SpecificUIActionItem definition

Name	Type	Description
SpecificUIActionItem	structure	
actionId	UInteger	Identifier for the action item that is specific to the UIP.
descriptor	String	A human readable string which provides information about the action.
enabled	Boolean	Indicates the state of the Action Item. “true” indicates the item is enabled. “false” indicates the item is disabled.
label	String	Label of the action item.

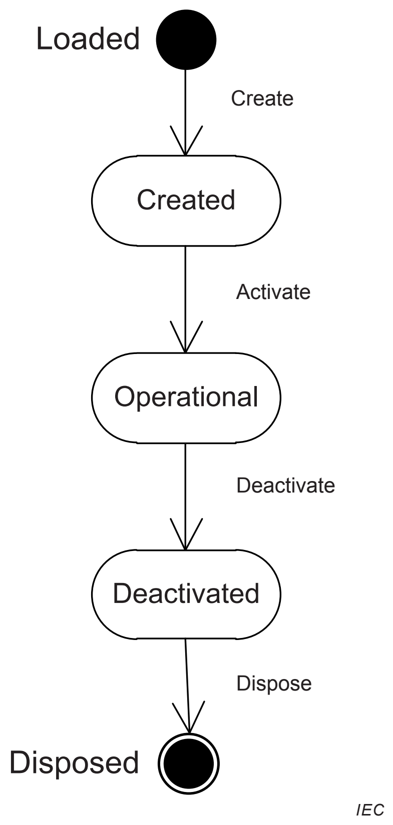
6.2 UIP instantiation rules

For each device instance there may be a maximum of two instances of the same UIP type, one for the offline version and one for the online version.

6.3 UIP state machine

6.3.1 States

Figure 7 shows the UIP state machine. If the UIP Services (see 6.1) trigger state changes, they are mentioned on the state machine edges.



NOTE Create and Dispose are technology dependent services described in IEC 62769-6. Create uses the StartElementName Property (see IEC 62769-5) of the UIP to create the appropriate UIP.

Figure 7 – UIP state machine

The UIP states are specified in Table 76.

Table 76 – UIP states

State	Description
Created	The initial state after the UIP instance is created by the FDI Client.
Operational	The normal execution state.
Deactivated	The final state before a UIP is disposed.

6.3.2 State transitions

The UIP state transitions are defined in Table 77.

Table 77 – UIP state transitions

Source state	Event	Destination state
Loaded	The FDI Client created the UIP instance.	Created
Created	Activate service has been called on UIP instance.	Operational
Operational	Deactivate service has been successfully called on UIP instance.	Deactivated
Deactivated	The FDI Client disposed the UIP instance.	Disposed

6.4 UIP permissions

A UIP is granted the following permissions with respect to operating system resources

- File system access (read/write) (mandatory).

- Printer access (mandatory – if a printer is available in the hosting environment).
- Network access (optional).
- ActiveX control usage (optional).

NOTE 1 The installation of ActiveX Controls is outside the scope of this standard.

Mandatory means that an FDI Client shall grant the permissions as they are granted by the operating system. This does not mean that a UIP has unrestricted access to the complete file system. The permissions for accessing the file system and printers are defined by the system configuration.

NOTE 2 The setting of permissions for operation system resources in a system configuration is outside the scope of this standard.

Optional means that an FDI Client can restrict the permissions compared to the permissions given by the system configuration. The means by which an FDI Client can restrict the network access and/or ActiveX control usage is specified in IEC 62769-6.

UIPs shall never block the Client, for instance by issuing synchronous calls to potentially block operating system resources. Worker threads are a common means for avoiding blocking situations.

6.5 UIP downloads from FDI Server

The following scenarios require an FDI Client to retrieve a UIP.

- The UID specifies a UIP in its XML element Plugin (see Annex A).
- The UIP opens another UIP with the OpenUserInterface service (see 5.2.2.3).

In all scenarios the FDI Client gets a UipId as identification for the UIP. A UipId is a UUID (universally unique identifier). A UIP may have several UIP Variants that differ in their platform and runtime support (see IEC 62769-4:2015, 5.3.3.2.2).

NOTE A UipId does not contain version information. Resolving a UipId to the appropriate version of the UIP is done by the FDI Server based on FDI Package information.

The same UipId may be resolved to different UIP versions for different Devices. At different points in time even for the same Device the same UipId may be resolved differently if a UIP update happened in the FDI Server in between.

With the UipId the FDI Client can retrieve the following information about all corresponding UIP Variants (see IEC 62769-5) from the FDI Server:

- RuntimeId and PlatformId
- UIPVariantVersion
- FDITechnologyVersion

For this the FDI Client calls the OPC UA service TranslateBrowsePathToNodeIds with the following list of relative names:

"UIPSet/<UipId>"

"UIPSet/<UipId >/RuntimeId"

"UIPSet/<UipId >/PlatformId"

"UIPSet/<UipId >/FDITechnologyVersion"

"UIPSet/<UipId >/Style"

"UIPSet/<UipId >/StartElementName"

"UIPSet/<UipId >/UIPVariantVersion"

The FDI Server returns arrays of NodeIds for each relative name. The number of entries in each array matches the number of UIP Variants for the UipId. Next the FDI Client can read the property values.

If multiple UIP Variants are available, the FDI Client chooses the most appropriate UIP Variant based on FDITechnologyVersion, RuntimeId, and PlatformId.

The FDI Client may maintain a cache of UIP Variants already downloaded from the FDI Server. If UipId, RuntimeId, PlatformId, FDITechnologyVersion and UIPVariantVersion of the chosen UIP Variant match with a UIP Variant in its UIP Variant cache, this UIP Variant can be used and no download from the FDI Server is required.

If no cache is maintained or no matching UIP Variant is found in the cache, the FDI Client shall read the FDITechnologyVersion (see IEC 62769-5) of the chosen UIP Variant. The parameter FDITechnologyVersion helps the FDI Client to determine whether it can support the execution of the UIP. The FDI Client shall execute a UIP of the same major version independently from the minor version or revision.

Finally the FDI Client can download the selected UIP from the FDI Server by reading the value of the UIP Variant.

An FDI Client may implement another sequence than the one described here. For example, it may check the FDITechnologyVersion first to determine whether it can run the UIP at all. The FDITechnologyVersion is the same for all UIP Variants of a specific version of a UIP.

The FDI Client may implement optimization strategies. For example, it may cache UIP versions (in addition to UIP Variant versions). If the version of the UIP is identical to the cached UIP version, there is no need to read the UIP Variant versions because they are required to be the same for an identical UIP version (see IEC 62769-4).

7 Actions

7.1 General

The EDD contained in the FDI Package as defined in IEC 62769-4 may have EDD methods. Some EDD methods may be exposed to the FDI Client as Actions and can be triggered by FDI Clients.

NOTE These Actions have no relationship to the EDD ACTION construct.

Actions are executed in the FDI Server. The Action state machine is defined in IEC 62769-3.

An FDI Client can invoke an Action by calling the OPC UA InvokeAction method (see IEC 62769-5). State changes of the corresponding Action state machine are sent to the FDI Client via the subscription mechanism (see IEC 62541-4). Additional data that may come with a state change are transferred as an XML document.

An Action may involve user interaction. The result of a user interaction is sent to the FDI Server with the RespondAction service (see IEC 62769-5). Data that are sent with this service are transferred as an XML document.

An Action can be aborted either by the Client of by the Server.

If the Server aborts an Action, it first sends an AbortingNotification. The Client shall reply with an AbortNotificationConfirmation but still continue processing ActionRequests. Once the AbortNotification was received, the Client shall disable the CancelButton. When the abort process is finished, the Server will send an AbortRequest to the Client. The Client replies with an AbortResponse and closes the ActionWindow.

An FDI Client may abort an Action as follows:

1) Call the AbortAction service (see IEC 62769-5)
2) Reply to an existing ActionRequest by sending a valid ActionResponse.
3) Continue processing of ActionRequests.
4) The Server sends an AbortingNotification once it has started the abort process. The Client shall reply with an AbortNotificationConfirmation but still continue processing ActionRequests. Once the AbortNotification was received, the Client shall disable the CancelButton.
5) The Action is aborted after an AbortingRequest is received from the Server. The Client shall reply with an AbortResponse and closes the ActionWindow.

Annex B contains an example of an EDD method being executed by an FDI Server and the resulting interaction with an FDI Client. It includes state diagrams and exchanged messages (XML snippets).

7.2 Sequence diagram

The sequence diagram shown in Figure 8 shows the client/server interaction of an Action call. This diagram assumes as a pre-condition that the Action can be started from a user interface shown by the FDI Client or by a UIP.

NOTE 1 The diagram shows UIDInterpreter, ActionWindow and AcknowledgeWindow as subsystems of the FDI Client. It also shows EDDMethodExecution as a subsystem of the FDI Server. This is for explanatory reasons only.

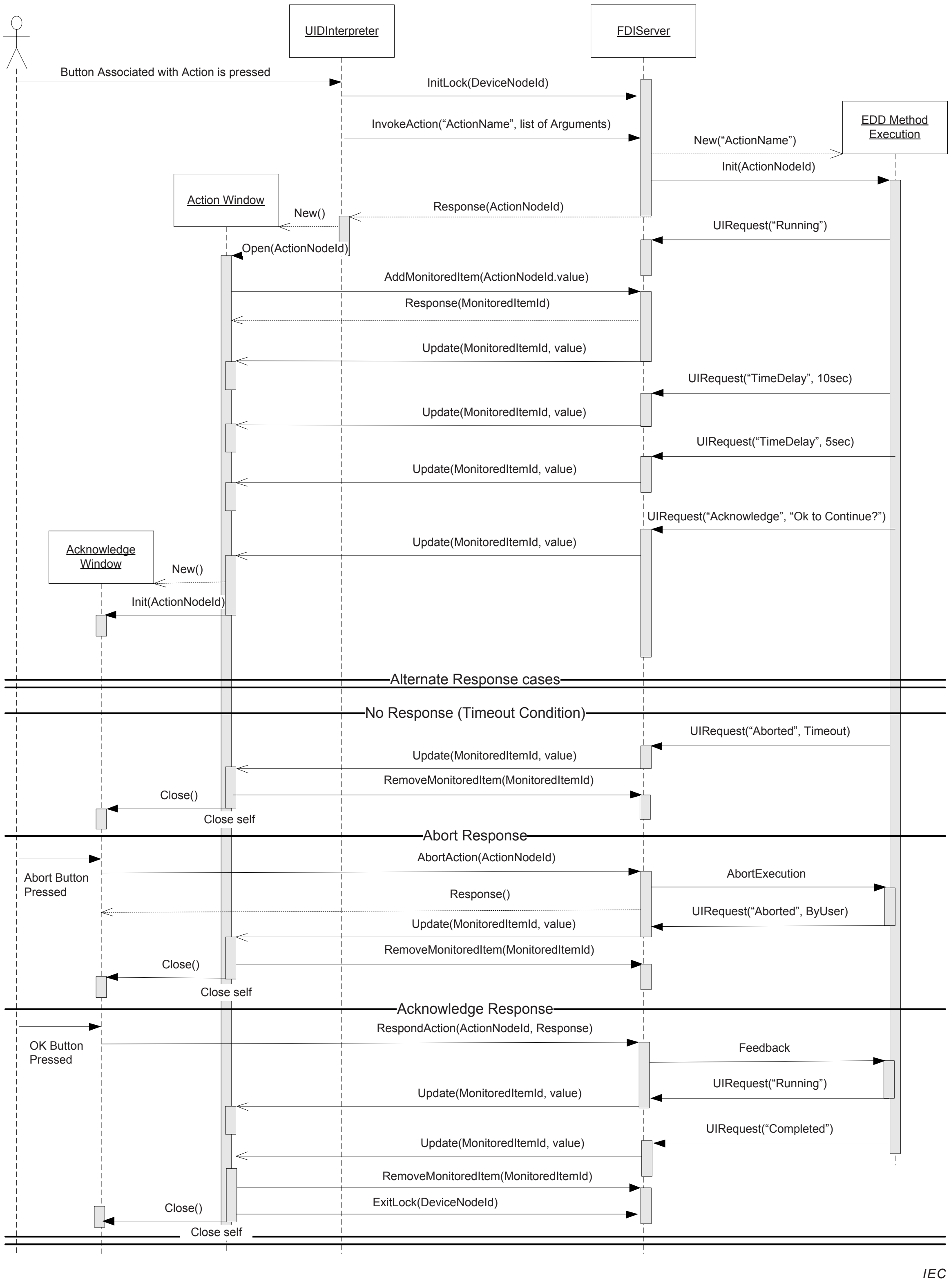

UIDInterpreter
FDIServer
EDD Method Execution
Action Window
Acknowledge Window
Button Associated with Action is pressed
InitLock(DeviceNodeId)
InvokeAction("ActionName", list of Arguments)
New("ActionName")
Init(ActionNodeId)
Response(ActionNodeId)
New()
UIRequest("Running")
Open(ActionNodeId)
AddMonitoredItem(ActionNodeId.value)
Response(MonitoredItemId)
Update(MonitoredItemId, value)
UIRequest("TimeDelay", 10sec)
Update(MonitoredItemId, value)
UIRequest("TimeDelay", 5sec)
Update(MonitoredItemId, value)
UIRequest("Acknowledge", "Ok to Continue?")
Update(MonitoredItemId, value)
New()
Init(ActionNodeId)
Alternate Response cases
No Response (Timeout Condition)
UIRequest("Aborted", Timeout)
Update(MonitoredItemId, value)
RemoveMonitoredItem(MonitoredItemId)
Close()
Close self
Abort Response
AbortAction(ActionNodeId)
Abort Button Pressed
AbortExecution
Response()
UIRequest("Aborted", ByUser)
Update(MonitoredItemId, value)
RemoveMonitoredItem(MonitoredItemId)
Close()
Close self
Acknowledge Response
RespondAction(ActionNodeId, Response)
OK Button Pressed
Feedback
UIRequest("Running")
Update(MonitoredItemId, value)
UIRequest("Completed")
Update(MonitoredItemId, value)
RemoveMonitoredItem(MonitoredItemId)
ExitLock(DeviceNodeId)
Close()
Close self
IEC

Figure 8 – FDI Action sequence diagram

The Action is initiated from the user interface. If not already locked, InitLock has to be called first. The FDI Client then calls the OPC UA InvokeAction method (see IEC 62769-5). The name of the Action to be invoked, as well as any Action arguments needed, is given as OPC UA method arguments.

NOTE 2 This description is identical whether the Action is initiated from a UIP or a UID. The only difference is that a UIP does not invoke the Action in the FDI Server directly but uses corresponding services in the DeviceAccess interface. The FDI Client then calls the Action on the FDI Server on behalf of the UIP.

NOTE 3 The interaction between FDI Client and FDI Server like the subscription to the ActionNodeId and the rendering of user interfaces (if requested during Action execution) is always under the responsibility of the FDI Client regardless of whether the Action has been initiated by a UID or UIP. Therefore a UIP need not be aware of the XML documents (as defined by the XML schema in Annex A) that are exchanged between FDI Client and FDI Server.

The FDI Server responds to the InvokeAction call by creating an Action execution state machine and responds to the FDI Client by returning an ActionNodeId that represents the state machine in the Information Model.

The FDI Server is responsible for running the state machine (as defined in IEC 62769-3) and updating its state in the Information Model using the provided ActionNodeId. The ActionNodeId is also used by the FDI Client to establish subscriptions with the FDI Server in order to monitor the Action execution. As the Action execution proceeds, the FDI Server updates the state machine and the corresponding Node in the Information Model. If there are any existing subscriptions for the ActionNodeId the FDI Server will process these changes and advise the FDI Client of the changes.

During the Action execution, operations such as notifications, read and write (also to other Nodes than the Node with the ActionNodeId) are not blocked.

The FDI Server begins an Action by first setting the state to Running and then starts to execute the EDD method. The FDI Client uses the OPC UA AddMonitoredItem service (see IEC 62541-4) to establish a subscription using the provided ActionNodeId. The FDI Server responds to the newly created subscription by advising the FDI Client of the current state of the Action state machine. In this sequence diagram the subscription was established before the Action execution reaches any notable state such as the TimeDelay state. Therefore the first notice to the FDI Client would indicate “Running”. If the Action had reached the TimeDelay state before the subscription was established the first notice to the FDI Client would indicate “TimeDelay” since this would be the current state.

The FDI Client shall not maintain an Action state machine.

NOTE 4 The information sent to the FDI Client combines state information and the last UI request. Therefore there is no race condition with respect to when the FDI Client adds the state Node as a monitored item. Even if the Action state machine has gone through several state transitions, the FDI Client need not know this. The standard OPC UA behaviour is that the current information is sent when a Node is added as monitored item.

When the FDI Server enters the TimeDelay state the ActionNodeId is updated at the beginning of the delay and at the end of the delay. There may also be intermediate updates regarding the length of the delay. The actual frequency of the intermediate updates is determined by the design of the FDI Server, typically every few seconds. All updates of the ActionNodeId will be submitted to the FDI Client via the subscription mechanism.

EXAMPLE 1 In Figure 8 the FDI Server has entered a time delay of 10 seconds and sends updates every 5 seconds.

When the FDI Server enters the WaitingForFeedback state the Information Model is updated including the user prompt. The state machine pauses the execution of the Action until a response from the user is provided or a timeout expires. The state machine change results in the FDI Client being provided a notification. The FDI Client detecting that the new state is WaitingForFeedback opens a message dialog presenting the prompt provided in the state information as well as a means to provide feedback.

EXAMPLE 2 In Figure 8 the FDI Client shows the buttons to “OK” the method or “Abort” the method.

The sequence diagram shows three alternate response cases.

Response case “Timeout”: If the user fails to respond to the prompt before the server timeout expires the FDI Server will abort the Action execution. The state in the Information Model is set to Aborted with an indication that the cause was a timeout condition. The Action execution terminates at this point but the FDI Server maintains the state Node and the final state until the subscriptions are terminated either by the FDI Client removing the monitored item or a general timeout of the FDI Client session with the FDI Server.

Response case “Abort”: If the user responds to the FDI Server with an abort response the method execution is aborted. The state in the Information Model is set to Aborted with an indication that the cause was a user action. The FDI Server maintains the state Node and the final state until the subscriptions are terminated either by the FDI Client removing the monitored item or a general timeout of the FDI Client session with the FDI Server. The state Node is also preserved if the Completed or Aborted state of the Action state machine is reached before the FDI Client established a subscription and added the state Node as a monitored item.

Response case “Positive feedback”: If the user responds to the FDI Server with positive feedback the Action execution sets the state in the Information Model back to Running and continues the normal execution of the Action.

EXAMPLE 3 In Figure 8 no further state changes occur before the end of the Action and therefore the next state change is to the final Completed state.

The state machine terminates after the final state is set. The FDI Server maintains the state Node and the final state until the subscriptions are terminated either by the FDI Client removing the monitored item or a general timeout of the FDI Client session with the FDI Server.

NOTE 5 No separate KeepAlive method is needed. The supervision of the session tells the FDI Server that the FDI Client is alive.

After the Action execution ended and the monitored item has been removed the lock can be removed.

7.3 FDI Action schema definition

The XML schema for the XML sent between the FDI Client and FDI Server is defined in Annex A.

NOTE The XML schema in Annex A also includes the definitions for UIDs.

ActionRequest is the root element of the XML documents that are exchanged from the FDI Server to the FDI Client during Action execution. These XML documents are sent to the FDI Client with the Update service (see IEC 62541-4). The mandatory part of these documents is the ActionState that specifies the current state of the corresponding Action state machine (see IEC 62769-3). Additional data is specified if the FDI Client has to show a user interface on behalf of the Action. The user interface style as well as content is specified in this additional data.

ActionResponse is the root element of the XML documents that are exchanged from FDI Client to FDI Server. These documents are used when the user provides feedback for a UI request. The Action state machine (as defined in IEC 62769-3) shall be in state WaitingForFeedback or WaitingForFeedbackA. The FDI Server sets the state back to Running or Aborting after receiving the user feedback. These XML documents are sent with the RespondAction service (as defined in IEC 62769-5).

Arguments for Actions are also specified by XML documents. The arguments are specified with the ListOfActionArguments type as name-value pairs.

8 User Interface Description (UID)

8.1 Overview

As defined in IEC 62769-5, top-level FunctionalGroup in the Information Model may contain a UIDescription (UID) Node or a set of UIPlugin (UIP) Nodes. The Value Attribute of a UID Node is a string, and the UID XML Schema defines the contents of that string. The XML schema is defined in Annex A.

NOTE 1 The XML schema in Annex A also includes the definitions for FDI Actions.

In addition to the UID Nodes exposed in the Information Model via FunctionalGroups, there may be non-browseable UID Nodes. As shown in Figure 9, non-browseable UID Nodes are linked to a parent UID Node via the NodePath attribute. The parent UID Node may be either a browseable or non-browseable Node.

The root element of the Value Attribute of a FunctionalGroup's UID Node shall be a Window, Dialog, Menu, or Table element. The root element of the Value Attribute of a non-browseable UID Node shall be a Window, Dialog, Page, Menu, or Table element.

The Value Attribute of a FunctionalGroup's UID Node shall contain enough information to allow the FDI Client to render the visible parts of the view. The FDI Server may omit those parts of the view that are not visible. In place of the omitted information the FDI Server shall provide a reference (via the NodePath attribute) to a non-browseable UID Node that contains the missing information. The FDI Client may use the specified NodePath to obtain the missing information from the FDI Server as it deems necessary.

NOTE 2 An example of a non-visible part of a window would be the second page of a tabbed dialog. The label of the second page is visible, but the content of the second page is not.

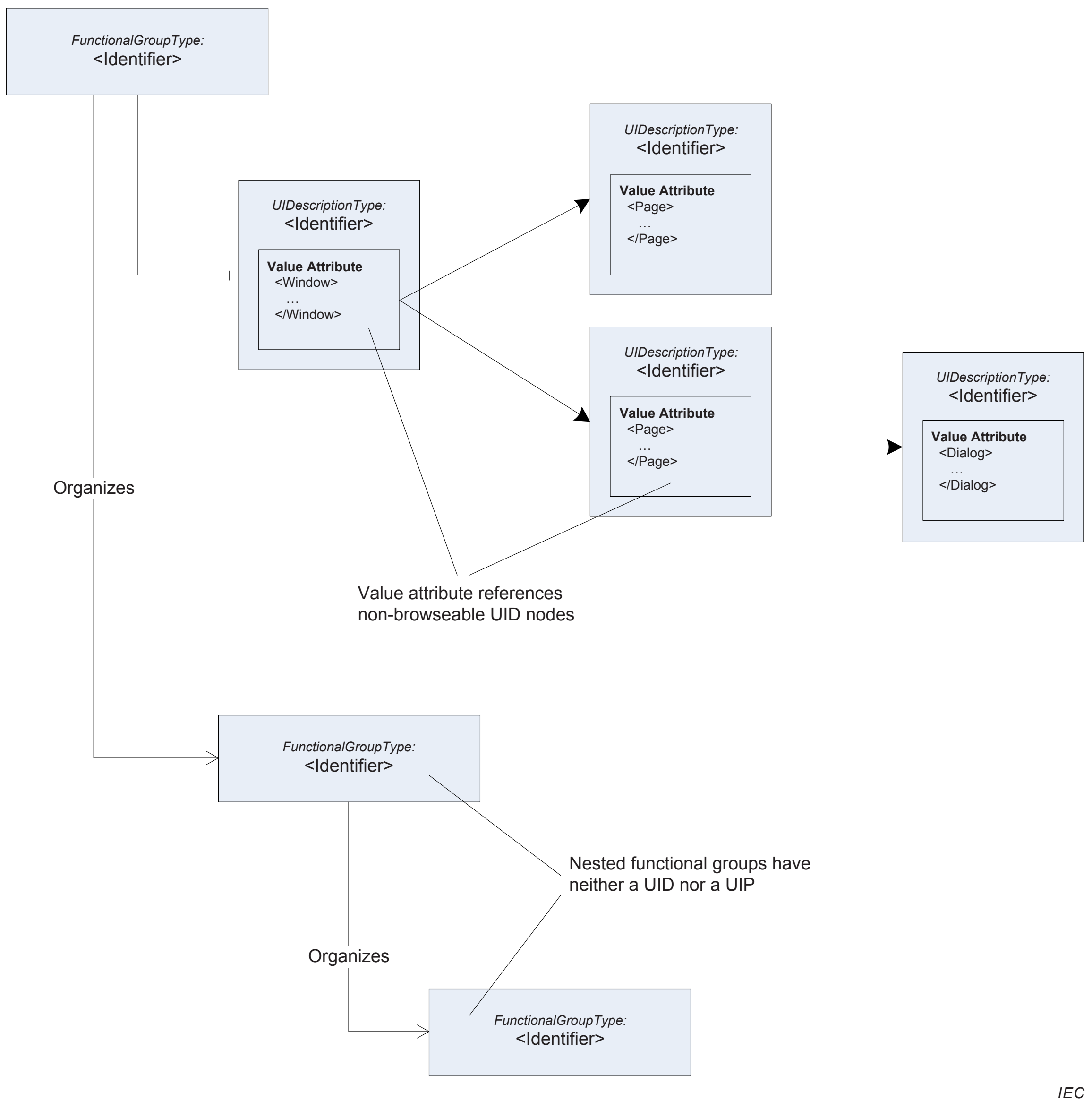


Figure 9 – User Interface Descriptions

The XML returned to the FDI Client from an FDI Server contains layout elements and content elements. Layout elements define the visual organization, positioning, and structure of the user interface. Content elements are the basic building blocks of the user interface.

NOTE 3 The XML only includes "valid" elements. New XML documents will be returned if validity changes.

The layout elements are

- ColumnBreak
- Dialog
- EditDisplay
- Group
- Menu
- Page
- RowBreak
- Table

- Window

The content elements are

- Action
- Chart
- Graph
- Grid
- Image
- Parameter
- Plugin
- Text

The algorithm for rendering these elements onto a computer screen is specified in IEC 61804-4.

NOTE 4 The definition of UIDs is heavily influenced by IEC 61804-3 and IEC 61804-4.

8.2 UID execution

Figure 10 illustrates an example of the sequence of steps used by an FDI Client to call up and execute a UIDescription (UID). This example assumes as a pre-condition that the FDI Client has established a session with the FDI Server, the user has navigated to a Device using some form of Information Model browsing or lookup, and the FDI Client is presenting the user with a list of FunctionalGroups. The example includes a sub UID that is referenced in the XML of the UID and that contains conditional content. The example illustrates the modification of a Parameter used by the sub UID to calculate the conditional content.

NOTE 1 The diagram shows UIDWindow, UIDInterpreter and Device Browser Window as subsystems of the FDI Client. It also shows UIDExecution as subsystem of the FDI Server. This is for explanatory reasons only.

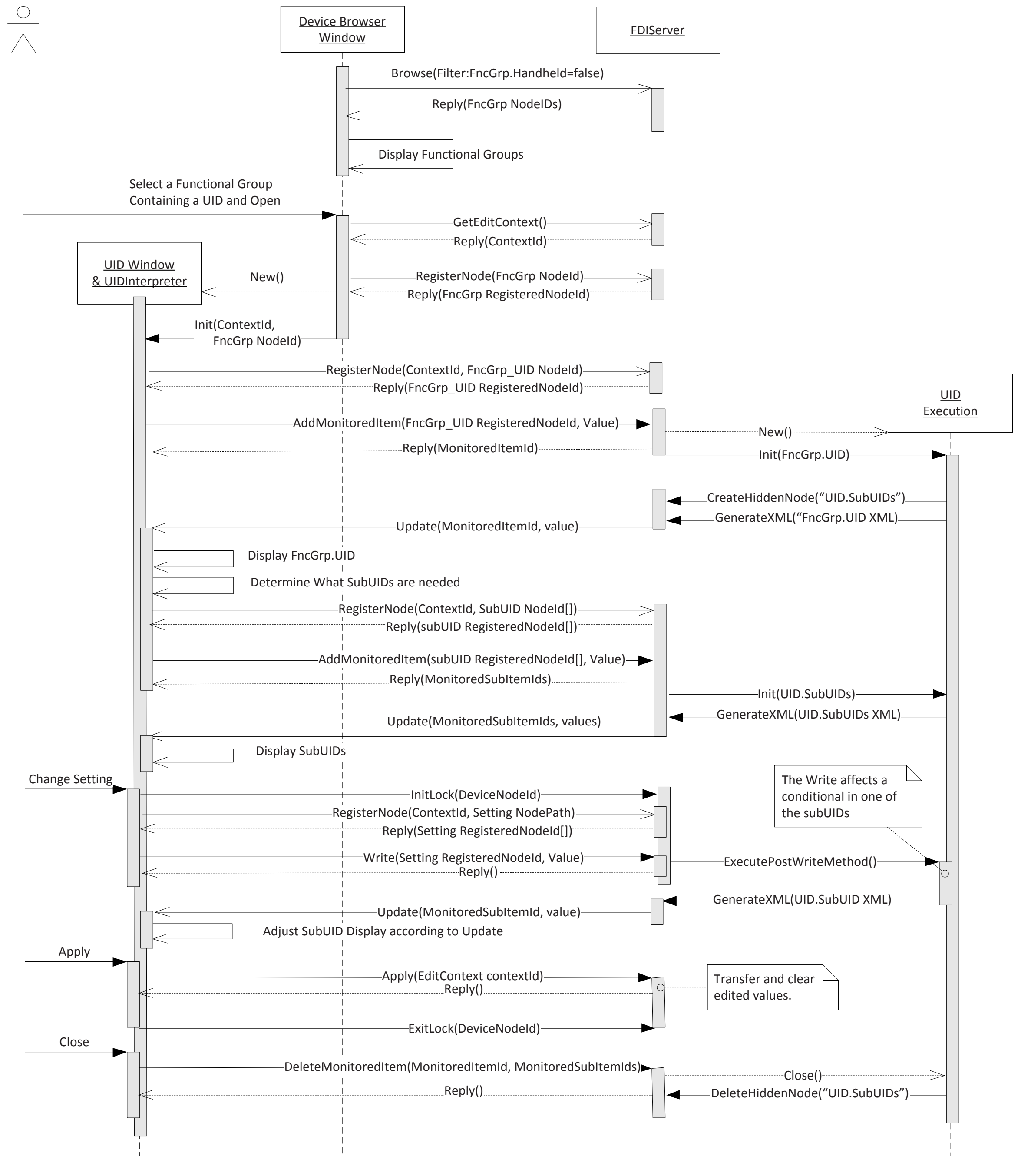

Device Browser Window
FDIServer
Browse(Filter:FncGrp.Handheld=false)
Reply(FncGrp NodeIDs)
Display Functional Groups
Select a Functional Group Containing a UID and Open
GetEditContext()
Reply(ContextId)
UID Window & UIDInterpreter
New()
RegisterNode(FncGrp NodeId)
Reply(FncGrp RegisteredNodeId)
Init(ContextId, FncGrp NodeId)
RegisterNode(ContextId, FncGrp_UID NodeId)
Reply(FncGrp_UID RegisteredNodeId)
UID Execution
AddMonitoredItem(FncGrp_UID RegisteredNodeId, Value)
New()
Reply(MonitoredItemId)
Init(FncGrp.UID)
CreateHiddenNode("UID.SubUIDs")
GenerateXML("FncGrp.UID XML)
Update(MonitoredItemId, value)
Display FncGrp.UID
Determine What SubUIDs are needed
RegisterNode(ContextId, SubUID NodeId[])
Reply(subUID RegisteredNodeId[])
AddMonitoredItem(subUID RegisteredNodeId[], Value)
Reply(MonitoredSubItemIds)
Init(UID.SubUIDs)
Update(MonitoredSubItemIds, values)
GenerateXML(UID.SubUIDs XML)
Display SubUIDs
Change Setting
InitLock(DeviceNodeId)
The Write affects a conditional in one of the subUIDs
RegisterNode(ContextId, Setting NodePath)
Reply(Setting RegisteredNodeId[])
Write(Setting RegisteredNodeId, Value)
ExecutePostWriteMethod()
Reply()
Update(MonitoredSubItemId, value)
GenerateXML(UID.SubUID XML)
Adjust SubUID Display according to Update
Apply
Apply(EditContext contextId)
Transfer and clear edited values.
Reply()
ExitLock(DeviceNodeId)
Close
DeleteMonitoredItem(MonitoredItemId, MonitoredSubItemIds)
Close()
Reply()
DeleteHiddenNode("UID.SubUIDs")
IEC

Figure 10 – User Interface Description sequence diagram

The sequence in Figure 10 begins with the user selecting one of the FunctionalGroups to open. The Device Browser Window initiates the opening of the UID associated with the selected FunctionalGroup by acquiring an EditContext and creating a new UID window and UID Interpreter that will be used to present the UID to the user. The newly created UID window is initialized by providing it with the EditContext and the NodeId of the selected FunctionalGroup in the EditContext. To retrieve this NodeId the RegisterNodesByRelativePath Method is called passing the BrowsePath of the FunctionalGroup and the EditContext as input.

Locking is required for the Write Service.

NOTE 2 The example therefore includes calls to lock (InitLock) and unlock (ExitLock) the Device.

The UID Window begins by establishing a subscription to the value of the UID contained in the referenced FunctionalGroup using the OPC UA AddMonitoredItem service in order to obtain the content (i.e., the XML representation) of the UID. The FDI Server responds to the subscription request by creating a UID Execution machine and initializing it by passing an internal identifier of the selected UID. The UID Execution machine interprets the UID definition, creates the referenced non-browseable sub UID Nodes in the Information Model and generates the XML representation of the top level UID. The generated XML representation is maintained in the Information Model to support the UID Window's subscribe. The NodePaths of the sub UIDs are included in the XML of the parent UIDs. Using the RegisterNodesByRelativePath Method the NodeSpecifiers are translated into NodeIds. The UID Window can use those NodeIds to create subsequent subscriptions. The FDI Server provides the content of the top level UID that was subscribed to by the UID Window.

The UID Window interprets the XML content of the updated UID value provided by the subscription and renders the user interface. It also collects the references to the sub UIDs and issues a second OPC UA AddMonitoredItem service request to include the sub UIDs in the subscription. The FDI Server responds to the AddMonitoredItem by initializing the sub UIDs Nodes resulting in the generation of XML representations of the sub UID content. The FDI Server updates the UID Window with the generated value of each of the sub UIDs included in the OPC UA AddMonitoredItem request.

NOTE 3 The example uses multiple sub UIDs (but is unspecific about how many). For example, AddMonitoredItem(UID.SubUIDs.value) adds the Nodes of all SubUIDs to the subscription.

The UID Window responds to the subscription updates by rendering the sub UIDs in the user interface. In addition to the sub UIDs, UIDs also contain NodePaths referencing parameters and actions. Using the RegisterNodesByRelativePath those NodePaths are translated to NodeIds and the UID Window uses the AddMonitoredItem service to subscribe to the values of the parameters. The FDI Server provides the values to the UID Window. The UID is now fully displayed and ready.

When the user starts making a change to a value the UID Window requests a lock using the InitLock Method.

There are different strategies when to acquire a lock. At the latest it needs to be acquired before any pre-edit actions are executed or the value is written to the FDI server.

If a pre-edit action is defined on the changed parameter the UID Window calls that action.

When the user finished changing the value the value is written to the EditContext using the Write service. If a post-edit action is defined on the changed paramter the UID Window calls that action.

The UID Execution determines that the value change results in the change of a conditional contained in one of the sub UIDs. The XML value of the sub UID that is affected is regenerated and the Information Model is updated. The FDI Server reacts to the Information Model update by notifying the UID Window of the change.

The UID Window processes the value change of the sub UID and makes the necessary adjustments to the user interface to reflect the change.

The user completes the example sequence by clicking on an apply button that instructs the UID Window to shut down. The UID Window calls the Apply Method for its EditContext and the changes are applied to the device. Any pre- and post-write actions are executed. The UID Window removes the subscriptions by calling the OPC UA DeleteMonitoredItem service call.

The FDI Server processes the OPC UA DeleteMonitoredItem call by removing the items from the subscription. Afterwards the UID Window calls the Discard Method on the EditContext and the FDI server removes the non-browseable sub UID Nodes and closes the UID Execution.

Annex A
(normative)

XML schema

A.1 General

The following is the XML schema definitions for UIDs as well as Actions. The types are listed in alphabetical order. The namespace xs: in the elements refers to the W3C XML Schema.

A.2 AbortRequestT

This type specifies a request sent from an FDI Server to an FDI Client when an Action is aborted. The FDI Client may inform the user about the reason the Action is aborting. Upon acknowledgement from the user, the FDI Client sends an ActionResponse back to the FDI Server.

The XML schema for an AbortRequestT type is:

```
<xs:complexType name="AbortRequestT">
   <xs:sequence>
      <xs:element name="Message" type="xs:string"/>
   </xs:sequence>
</xs:complexType>
```

The elements of an AbortRequestT type are described in Table A.1.

Table A.1 – Elements of AbortRequestT

Element	Description
Message	This required element specifies the reason the action has been aborted.

A.3 AccessT

This type specifies whether the access shall be ONLINE or OFFLINE.

The XML schema for an AccessT enumeration type is:

```
<xs:simpleType name="AccessT">
   <xs:restriction base="xs:string">
      <xs:enumeration value="ONLINE"/>
      <xs:enumeration value="OFFLINE"/>
   </xs:restriction>
</xs:simpleType>
```

The enumeration values of an AccessT enumeration type are described in Table A.2.

Table A.2 – Enumerations of AccessT

Enumeration	Description
ONLINE	The access shall be done ONLINE.
OFFLINE	The access shall be done OFFLINE.

A.4 AcknowledgementRequestT

This type specifies a request sent from an FDI Server to an FDI Client when the user needs to acknowledge a condition or state within an Action. Upon acknowledgement from the user, the FDI Client sends an AcknowledgementResponse back to the FDI Server. An AcknowledgementRequest shall only be used to acknowledge normal operating conditions. An AbortRequest shall be used to acknowledge a condition that leads to the Action being aborted.

The XML schema for an AcknowledgementRequestT type is:

```
<xs:complexType name="AcknowledgementRequestT">
   <xs:sequence>
      <xs:element name="Message" type="xs:string"/>
   </xs:sequence>
</xs:complexType>
```

The elements of an AcknowledgementRequestT type are described in Table A.3.

Table A.3 – Elements of AcknowledgementRequestT

Element	Description
Message	This required element specifies the condition the user is being asked to acknowledge in the form of a message to the user.

A.5 ActionListT

This type specifies a list of Action elements.

The XML schema for an ActionListT type is:

```
<xs:complexType name="ActionListT">
   <xs:sequence maxOccurs="unbounded">
      <xs:element name="Action" type="clnt:ActionT"/>
   </xs:sequence>
</xs:complexType>
```

The elements of an ActionListT type are described in Table A.4.

Table A.4 – Elements of ActionListT

Element	Description
Action	An element of the list.

A.6 AbortingNotificationT

This type is used in the ActionRequest element to notify the client that the action has been aborted on the server.

The XML schema for an AbortingNotificationT type is:

```
<xs:complexType name="AbortingNotificationT"/>
```

A.7 ActionRequestT

This type specifies an action request from the FDI server.

The XML schema for an ActionRequestT type is:

```
<xs:complexType name="ActionRequestT">
   <xs:sequence>
      <xs:element name="EditContext" type="xs:string"/>
      <xs:choice>
         <xs:element name="AbortingNotification" type="clnt:AbortingNotificationT"/>
         <xs:element name="AcknowledgementRequest"
            type="clnt:AcknowledgementRequestT"/>
         <xs:element name="AbortRequest" type="clnt:AbortRequestT"/>
         <xs:element name="UIDRequest" type="clnt:UidRequestT"/>
         <xs:element name="SelectionRequest" type="clnt:SelectionRequestT"/>
         <xs:element name="InputRequest" type="clnt:InputRequestT"/>
         <xs:element name="ParameterInputRequest"
            type="clnt:ParameterInputRequestT"/>
         <xs:element name="InfoRequest" type="clnt:InfoRequestT"/>
         <xs:element name="DelayMessageRequest" type="clnt:DelayMessageRequestT"/>
      </xs:choice>
   </xs:sequence>
</xs:complexType>
```

The elements of an ActionRequestT type are described in Table A.5.

Table A.5 – Elements of ActionRequestT

Element	Description
EditContext	This element specifies the EditContext to be used when editing Variables for this Action.
AbortingNotification	This element specifies that the action has been aborted on the server. To ensure correct abort processing, the User Interface shall not allow the cancellation of further action requests.
AcknowledgementRequest	This optional element specifies a request from an FDI Server to an FDI Client for the user to acknowledge a condition. The FDI Client will not respond to the FDI Server until the user acknowledged the condition.
AbortRequest	This optional element specifies a request from an FDI Server to an FDI Client to notify the user the Action is aborting.
UIDRequest	This optional element specifies a request from an FDI Server to an FDI Client to display complex user interface.
SelectionRequest	This optional element specifies a request from an FDI Server to an FDI Client for the user to make a selection from a list of possible alternatives
InputRequest	This optional element specifies a request from an FDI Server to an FDI Client for the user to edit data.
ParameterInputRequest	This optional element specifies a request from an FDI Server to an FDI Client for the user to edit a parameter.

Element	Description
InfoRequest	This optional element specifies a request from an FDI Server to an FDI Client for a message to be displayed to the user. The message does not require user acknowledgement and the FDI Client will respond immediately after displaying the message.
DelayMessageRequest	This type specifies a request sent from an FDI Server to an FDI Client when a delay is requested within an Action. The FDI Client informs the user of the reason for and duration of the delay. The FDI Client times the delay and sends a DelayMessageResponse back to the FDI Server after the time has elapsed.

A.8 ActionResponseT

This type specifies an action response to the FDI Server.

The XML schema for an ActionResponseT type is:

```
<xs:complexType name="ActionResponseT">
   <xs:choice>
      <xs:element name="AbortingNotificationConfirmation" type="clnt:ResponseT"/>
      <xs:element name="AcknowledgementResponse" type="clnt:ResponseT"/>
      <xs:element name="AbortResponse" type="clnt:ResponseT"/>
      <xs:element name="UidResponse" type="clnt:UidResponseT"/>
      <xs:element name="SelectionResponse" type="clnt:SelectionResponseT"/>
      <xs:element name="InputResponse" type="clnt:InputResponseT"/>
      <xs:element name="ParameterInputResponse" type="clnt:ResponseT"/>
      <xs:element name="InfoResponse" type="clnt:ResponseT"/>
      <xs:element name="DelayMessageResponse" type="clnt:ResponseT"/>
   </xs:choice>
</xs:complexType>
```

The elements of an ActionResponseT type are described in Table A.6.

Table A.6 – Elements of ActionResponseT

Element	Description
AbortingNotificationConfirmation	This optional element specifies the response of an FDI Client to an AbortingNotification from an FDI Server.
AcknowledgementResponse	This optional element specifies the response of an FDI Client to an AcknowledgementRequest from an FDI Server.
AbortResponse	This optional element specifies the response of an FDI Client to an AbortRequest from an FDI Server.
UidResponse	This optional element specifies the response of an FDI Client to an UIDRequest from an FDI Server.
SelectionResponse	This optional element specifies the response of an FDI Client to a SelectionRequest from an FDI Server.
InputResponse	This optional element specifies the response of an FDI Client to an InputRequest from an FDI Server.
ParameterInputResponse	This optional element specifies the response of an FDI Client to an ParameterInputRequest from an FDI Server.
InfoResponse	This optional element specifies the response of an FDI Client to an InfoRequest from an FDI Server.
DelayMessageResponse	This optional element specifies the response of an FDI Client to a DelayMessageRequest from an FDI Server.

A.9 ActionT

This type specifies an Action, which is a sequence of steps that requires collaboration between an FDI Client and an FDI Server.

The XML schema for an ActionT type is:

```
<xs:complexType name="ActionT">
   <xs:complexContent>
      <xs:extension base="clnt:UiElementT">
         <xs:sequence>
            <xs:element name="Name" type="xs:string"/>
            <xs:element name="Access" type="clnt:AccessT" minOccurs="0"/>
            <xs:element name="Class" type="clnt:ActionClassT" minOccurs="0"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The elements of an ActionT type are described in Table A.7.

Table A.7 – Elements of ActionT

Element	Description
Name	This required element specifies the name of the Action, which may be passed to the InvokeAction method of a UID Node in an FDI Server.
Access	This optional element specifies whether the action shall be used ONLINE or OFFLINE.
Class	This optional element specifies the EDD CLASS attribute of the action.

A.10 AxisListT

This type specifies list of named axis elements of a graph or a chart.

The XML schema for an AxisListT type is:

```
<xs:complexType name="AxisListT">
   <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:element name="Axis" type="clnt:AxisT"/>
   </xs:sequence>
</xs:complexType>
```

The elements of a AxisListT type are described in Table A.8.

Table A.8 – Elements of AxisListT

Element	Description
Axis	An element of the AxisListT.

A.11 AxisT

This type specifies an axis of a graph or a chart.

The XML schema for an AxisT type is:

```
<xs:complexType name="AxisT">
   <xs:complexContent>
      <xs:extension base="clnt:LabelHelpT">
         <xs:sequence>
            <xs:element name="MaximumValue" type="clnt:VariantT" minOccurs="0"/>
            <xs:element name="MinimumValue" type="clnt:VariantT" minOccurs="0"/>
            <xs:element name="DisplayedRange" minOccurs="0">
               <xs:complexType>
                  <xs:attribute name="NodePathViewMinimum" type="xs:string"
                     use="required"/>
                  <xs:attribute name="NodePathViewMaximum" type="xs:string"
                     use="required"/>
               </xs:complexType>
            </xs:element>
            <xs:element name="Scaling" type="clnt:ScalingT" default="Linear"
               minOccurs="0"/>
            <xs:element name="Unit" type="xs:string" minOccurs="0"/>
         </xs:sequence>
         <xs:attribute name="Name" type="xs:string" use="required"/>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The attributes of an AxisT type are described in Table A.9.

Table A.9 – Attributes of AxisT

Attribute	Description
Name	Unique name of axis in the context of Chart/Graph that this axis is present.

The elements of an AxisT type are described in Table A.10.

Table A.10 – Elements of AxisT

Element	Description
MaximumValue	This required element specifies the largest value that lies within the bounds of the axis.
MinimumValue	This required element specifies the smallest value that lies within the bounds of the axis.
DisplayedRange	Sn1p1t Sn1p1t
Scaling	This optional element specifies how the values should be scaled. The default is Linear.
Unit	This optional element specifies the engineering unit of the axis.

A.12 BitEnumerationItemListT

This type specifies the content of a bit enumeration

The XML schema for a BitEnumerationItemListT type is:

```
<xs:complexType name="BitEnumerationItemListT">
   <xs:sequence maxOccurs="unbounded">
```

```
      <xs:element name="BitEnumerationItem" type="clnt:BitEnumerationItemT"/>
   </xs:sequence>
</xs:complexType>
```

The elements of a BitEnumerationItemListT type are described in Table A.11.

Table A.11 – Elements of BitEnumerationItemListT

Element	Description
BitEnumerationItem	An element of the bit enumeration.

A.13 BitEnumerationItemT

This type specifies the meaning of a single bit of a bitmapped value.

The XML schema for a BitEnumerationItemT type is:

```
<xs:complexType name="BitEnumerationItemT">
   <xs:complexContent>
      <xs:extension base="clnt:LabelHelpT">
         <xs:sequence>
            <xs:element name="Value" type="xs:unsignedLong"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The elements of a BitEnumerationItemT type are described in Table A.12.

Table A.12 – Elements of BitEnumerationItemT

Element	Description
Value	This required element specifies the bit as a bit mask, i.e., 0x1 specifies the least significant bit, 0x2 specifies the second least significant bit, 0x4 specifies the third most significant bit, and so on.

A.14 ButtonListT

This type specifies a list of Button elements. The UID response contains the 0-based index of the selected button.

The XML schema for a ButtonListT type is:

```
<xs:complexType name="ButtonListT">
   <xs:sequence maxOccurs="unbounded">
      <xs:element name="Button" type="clnt:LabelT"/>
   </xs:sequence>
</xs:complexType>
```

The elements of a ButtonListT type are described in Table A.13.

Table A.13 – Elements of ButtonListT

Element	Description
Button	An element of the button list.

A.15 ChartT

This type specifies a chart that graphically displays data from a device. The data is read from the device periodically and continuously. As new data arrives, it is displayed.

The XML schema for a ChartT type is:

```
<xs:complexType name="ChartT">
   <xs:complexContent>
      <xs:extension base="clnt:UiElementSizeableT">
         <xs:sequence>
            <xs:element name="Length" type="xs:nonNegativeInteger" default="600000"
               minOccurs="0"/>
            <xs:element name="Type" type="clnt:ChartTypeT" default="Strip"
               minOccurs="0"/>
            <xs:element name="CycleTime" type="xs:nonNegativeInteger" default="1000"
               minOccurs="0"/>
            <xs:element name="AxisList" type="clnt:AxisListT">
               <xs:key name="AxisKey">
                  <xs:selector xpath="clnt:Axis"/>
                  <xs:field xpath="@Name"/>
               </xs:key>
            </xs:element>
            <xs:element name="SourceList" type="clnt:SourceListT"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The elements of a ChartT type are described in Table A.14.

Table A.14 – Elements of ChartT

Element	Description
Length	This optional element specifies the length of time, in milliseconds, that is displayed by the chart. The number of samples displayed by a chart can be calculated by dividing the Length by the CycleTime. The default is 600 000.
Type	This optional element specifies the type of the chart. The default is Strip.
CycleTime	This optional element specifies the rate, in milliseconds, at which data on the chart is updated. The default is 1 000.
AxisList	This required element specifies the vertical axis data displayed by the chart.
SourceList	This required element specifies the data displayed by the chart.

A.16 ChartTypeT

This type specifies the general appearance and behavior of a chart.

The XML schema for a ChartTypeT enumeration type is:

```
<xs:simpleType name="ChartTypeT">
   <xs:restriction base="xs:string">
      <xs:enumeration value="Gauge"/>
      <xs:enumeration value="HorizontalBar"/>
      <xs:enumeration value="Scope"/>
      <xs:enumeration value="Strip"/>
      <xs:enumeration value="Sweep"/>
      <xs:enumeration value="VerticalBar"/>
   </xs:restriction>
</xs:simpleType>
```

The enumeration values of a ChartTypeT enumeration type are described in Table A.15.

Table A.15 – Enumerations of ChartTypeT

Enumeration	Description
Gauge	A single source value is displayed as a gauge, which is a graphical representation similar to the fuel gauge in an automobile.
HorizontalBar	The source values are displayed as horizontal bars.
Scope	The source values are displayed as a curve moving from left to right. When the source values reach the far right of the display area, the display area is erased and the new source values are displayed beginning at the far left.
Strip	The source values are displayed as a curve moving from left to right. When the source values reach the far right of the display area, the display area is scrolled with the source values on the far left being removed and the new source values being displayed on the right.
Sweep	The source values are displayed as a curve moving from left to right. When the source values reach the far right of the display area, the new source values are displayed beginning at the far left, but unlike Scope only the portion of the display area needed to display the new source values is erased.
VerticalBar	The source values are displayed as vertical bars.

A.17 ColorNameT

This type specifies the name of a predefined color.

The XML schema for a ColorNameT enumeration type is:

```
<xs:simpleType name="ColorNameT">
   <xs:restriction base="xs:string">
      <xs:enumeration value="Agua"/>
      <xs:enumeration value="Black"/>
      <xs:enumeration value="Blue"/>
      <xs:enumeration value="Fuchsia"/>
      <xs:enumeration value="Gray"/>
      <xs:enumeration value="Green"/>
      <xs:enumeration value="Lime"/>
      <xs:enumeration value="Maroon"/>
      <xs:enumeration value="Navy"/>
      <xs:enumeration value="Olive"/>
      <xs:enumeration value="Purple"/>
      <xs:enumeration value="Red"/>
      <xs:enumeration value="Silver"/>
      <xs:enumeration value="Teal"/>
      <xs:enumeration value="White"/>
      <xs:enumeration value="Yellow"/>
   </xs:restriction>
</xs:simpleType>
```

The enumeration values of a ColorNameT enumeration type are described in Table A.16.

Table A.16 – Enumerations of ColorNameT

Enumeration	Description
Agua	RGB value #00FFFF.
Black	RGB value #000000.
Blue	RGB value #0000FF.
Fuchsia	RGB value #FF00FF.
Gray	RGB value #808080.
Green	RGB value #008000.
Lime	RGB value #00FF00.
Maroon	RGB value #800000.
Navy	RGB value #000080.
Olive	RGB value #808000.
Purple	RGB value #800080.
Red	RGB value #FF0000.
Silver	RGB value #C0C0C0.
Teal	RGB value #008080.
White	RGB value #FFFFFF.
Yellow	RGB value #FFFF00.

A.18 ColorT

This type specifies a color, either a name or an RGB value.

The XML schema for a ColorT type is:

```
<xs:simpleType name="ColorT">
   <xs:union memberTypes="clnt:ColorNameT clnt:ColorValueT"/>
</xs:simpleType>
```

A.19 ColorValueT

This type specifies an RBG value consisting of a hash character (#) followed by three or six hexadecimal digits. If three hexadecimal digits are specified, each character is repeated, for example, #F0F is equivalent to #FF00FF.

The XML schema for a ColorValueT type is:

```
<xs:simpleType name="ColorValueT">
   <xs:restriction base="xs:string">
      <xs:pattern value="#[0-9a-fA-F]{3}"/>
      <xs:pattern value="#[0-9a-fA-F]{6}"/>
   </xs:restriction>
</xs:simpleType>
```

A.20 ColumnBreakT

This type specifies a column break.

The XML schema for a ColumnBreakT type is:

```
<xs:complexType name="ColumnBreakT"/>
```

A.21 DateTimeDataT

This type specifies the data type of a date/time value.

The XML schema for a DateTimeDataT enumeration type is:

```
<xs:simpleType name="DateTimeDataT">
   <xs:restriction base="xs:string">
      <xs:enumeration value="Date"/>
      <xs:enumeration value="Time"/>
      <xs:enumeration value="DateTime"/>
      <xs:enumeration value="Duration"/>
      <xs:enumeration value="TimeValue"/>
   </xs:restriction>
</xs:simpleType>
```

The enumeration values of a DateTimeDataT enumeration type are described in Table A.17.

Table A.17 – Enumerations of DateTimeDataT

Enumeration	Description
Date	A value that represents a date without a time.
Time	A value that represents a time without a date.
DateTime	A value that represents a date and time.
Duration	A value that represents a length of time.
TimeValue	A value that represents a time of the day.

A.22 DelayMessageRequestT

This type specifies a request sent from an FDI Server to an FDI Client when a delay occurs within an Action. The FDI Client informs the user of the reason for the delay and expected duration of the delay. The FDI Client then sends an ActionResponse back to the FDI Server. The FDI Server may send DelayMessageRequests frequently with updated information regarding the length of the delay. When the delay is finished, the FDI Server sends a DelayMessageRequest with the SecondsToWait element set to zero.

The XML schema for a DelayMessageRequestT type is:

```
<xs:complexType name="DelayMessageRequestT">
   <xs:sequence>
      <xs:element name="Message" type="xs:string"/>
      <xs:element name="SecondsToWait" type="xs:unsignedLong"/>
   </xs:sequence>
</xs:complexType>
```

The elements of a DelayMessageRequestT type are described in Table A.18.

Table A.18 – Elements of DelayMessageRequestT

Element	Description
Message	This required element specifies the message to be displayed to the user.
SecondsToWait	This required element specifies the number of seconds the delay is expected to last.

A.23 DiagramLineT

This type specifies an abstract source

The XML schema for a DiagramLineT type is:

```
<xs:complexType name="DiagramLineT" abstract="true">
   <xs:complexContent>
      <xs:extension base="clnt:UiElementT">
         <xs:sequence>
            <xs:element name="Emphasis" type="xs:boolean" default="false"
               minOccurs="0"/>
            <xs:element name="LineColor" type="clnt:ColorT" minOccurs="0"/>
            <xs:element name="LineType" type="clnt:LineTypeT" minOccurs="0"/>
            <xs:element name="VerticalAxis" minOccurs="0">
               <xs:complexType>
                  <xs:attribute name="AxisRef" type="xs:string"/>
               </xs:complexType>
            </xs:element>
            <xs:element name="InitActionList" type="clnt:ActionListT" minOccurs="0"/>
            <xs:element name="RefreshActionList" type="clnt:ActionListT"
               minOccurs="0"/>
            <xs:element name="ExitActionList" type="clnt:ActionListT" minOccurs="0"/>
         </xs:sequence>
         <xs:attribute name="Name" type="xs:string" use="required"/>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The attributes of a DiagramLineT type are described in Table A.19.

Table A.19 – Attributes of DiagramLineT

Attribute	Description
Name	Unique name of DiagramlineT in the context of Chart/Graph that this DiagramlineT is present.

The elements of a DiagramLineT type are described in Table A.20.

Table A.20 – Elements of DiagramLineT

Element	Description
Emphasis	This optional element specifies whether or not this data should be graphically emphasized. The default is false.
LineColor	This optional element specifies the color in which to display the data. The default is determined by the FDI Client.
LineType	This optional element specifies the type of line to be displayed. Every data with the same LineType should be displayed in the same fashion (line pattern, line thickness, etc).
VerticalAxis	This optional element specifies appearance of the vertical axis. Sn1p1t
InitActionList	This optional element specifies the Actions to be executed before the data is displayed.
RefreshActionList	This optional element specifies the Actions to be executed after the data is read from the device but before it is displayed.
ExitActionList	This optional element specifies the Actions to be executed when the graph or chart containing the data is closed.

A.24 EnumerationItemListT

This type specifies the content of an enumeration.

The XML schema for an EnumerationItemListT type is:

```
<xs:complexType name="EnumerationItemListT">
   <xs:sequence maxOccurs="unbounded">
      <xs:element name="EnumerationItem" type="clnt:EnumerationItemT"/>
   </xs:sequence>
</xs:complexType>
```

The elements of an EnumerationItemListT type are described in Table A.21.

Table A.21 – Elements of EnumerationItemListT

Element	Description
EnumerationItem	An element of the enumeration.

A.25 EnumerationItemT

This type specifies one of the possible values of an enumerated value.

The XML schema for an EnumerationItemT type is:

```
<xs:complexType name="EnumerationItemT">
   <xs:complexContent>
      <xs:extension base="clnt:LabelHelpT">
         <xs:sequence>
            <xs:element name="Value" type="xs:unsignedLong"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The elements of an EnumerationItemT type are described in Table A.22.

Table A.22 – Elements of EnumerationItemT

Element	Description
Value	This required element specifies the unique identifier of the list entry in the context of the parent list element.

A.26 FormatSpecifierT

This type specifies the allowed format specifier of parameters (ANSI C compatible).

The XML schema for a FormatSpecifierT type is:

```
<xs:simpleType name="FormatSpecifierT">
   <xs:restriction base="xs:string">
      <xs:pattern value="%?[#-0+ ]*\d*(\.\d*)?[hlL]?[dioxXucsfeEgGpn%]"/>
   </xs:restriction>
</xs:simpleType>
```

A.27 GraphT

This type specifies a graph that graphically displays a finite data set from a device.

The XML schema for a GraphT type is:

```
<xs:complexType name="GraphT">
   <xs:complexContent>
      <xs:extension base="clnt:UiElementSizeableT">
         <xs:sequence>
            <xs:element name="CycleTime" type="xs:nonNegativeInteger" default="0"
               minOccurs="0"/>
            <xs:element name="AxisList" type="clnt:AxisListT">
               <xs:key name="GraphAxisKey">
                  <xs:selector xpath="clnt:Axis"/>
                  <xs:field xpath="@Name"/>
               </xs:key>
            </xs:element>
            <xs:element name="HorizontalAxis" minOccurs="0">
               <xs:complexType>
                  <xs:attribute name="AxisRef" type="xs:string"/>
               </xs:complexType>
            </xs:element>
            <xs:element name="WaveformList" type="clnt:WaveformListT"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The elements of a GraphT type are described in Table A.23.

Table A.23 – Elements of GraphT

Element	Description
CycleTime	This optional element specifies the rate, in milliseconds, at which data set is re-read from the device and re-displayed. If the CycleTime is 0, the graph is never refreshed. The default is 0.
AxisList	This required element contains all the axes referred from graph or waveform.
HorizontalAxis	Sn1p1t
WaveformList	This required element specifies the waveforms displayed on the graph.

A.28 GridT

This type specifies a grid that displays data in a table-like grid.

The XML schema for a GridT type is:

```
<xs:complexType name="GridT">
   <xs:complexContent>
      <xs:extension base="clnt:UiElementSizeableT">
         <xs:sequence>
            <xs:element name="Handling" type="clnt:HandlingT" default="rw"
               minOccurs="0"/>
            <xs:element name="Orientation" type="clnt:OrientationT"
               default="Vertical" minOccurs="0"/>
            <xs:element name="VectorList" type="clnt:VectorListT"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The elements of a GridT type are described in Table A.24.

Table A.24 – Elements of GridT

Element	Description
Handling	This optional element specifies whether or not the data within the grid may be modified. The default is ReadWrite.
Orientation	This optional element specifies whether the vectors are displayed horizontally or vertically. The default is Vertical.
VectorList	This required element specifies the data displayed within the grid.

A.29 HandlingT

This type specifies how an item may be accessed (ro = read only, wo = write only, rw = read and write).

The XML schema for a HandlingT enumeration type is:

```
<xs:simpleType name="HandlingT">
   <xs:restriction base="xs:string">
      <xs:enumeration value="rw"/>
      <xs:enumeration value="wo"/>
      <xs:enumeration value="ro"/>
   </xs:restriction>
</xs:simpleType>
```

The enumeration values of a HandlingT enumeration type are described in Table A.25.

Table A.25 – Enumerations of HandlingT

Enumeration	Description
rw	The item may only be read.
wo	The item may only be written.
ro	The item may be read or written.

A.30 ImageT

This type specifies a visual entity such as a picture or illustration.

The XML schema for an ImageT type is:

```
<xs:complexType name="ImageT">
   <xs:complexContent>
      <xs:extension base="clnt:UiElementT">
         <xs:sequence>
            <xs:element name="Inline" type="xs:boolean" default="false"
               minOccurs="0"/>
            <xs:element name="AlignLeft" type="xs:boolean" default="false"
               minOccurs="0"/>
            <xs:element name="AlignRight" type="xs:boolean" default="false"
               minOccurs="0"/>
            <xs:element name="Link" minOccurs="0">
               <xs:complexType>
                  <xs:choice>
                     <xs:element name="Menu" type="clnt:MenuReferenceT"/>
                     <xs:element name="Plugin" type="clnt:PluginT"/>
                     <xs:element name="Action" type="clnt:ActionT"/>
                  </xs:choice>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
         <xs:attribute name="NodePath" type="xs:string" use="required"/>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The attributes of an ImageT type are described in Table A.26.

Table A.26 – Attributes of ImageT

Attribute	Description
NodePath	This is a mandatory attribute which gives the qualified path of the Node in the Device Model.

The elements of an ImageT type are described in Table A.27.

Table A.27 – Elements of ImageT

Element	Description
Inline	If the image is in a window, dialog, page or group and the inline qualifier is not specified, the image should span with the width of the MENU; otherwise, the image should be displayed inline with other layoutitems of the itemlist.
AlignLeft	If the image is in a window, dialog, page or group and the inline qualifier is not specified, the image should span with the width of the MENU; otherwise, the image should be displayed inline with other layoutitems of the itemlist.
AlignRight	If the image is in a window, dialog, page or group and the inline qualifier is not specified, the image should span with the width of the MENU; otherwise, the image should be displayed inline with other layoutitems of the itemlist.
Link	This optional element specifies a link to Window, Dialog, Menu, Table, or Action related to the image.

A.31 InfoRequestT

This type specifies a request sent from an FDI Server to an FDI Client when a message needs to be displayed to the user needs during an Action. Unlike an AcknowledgementRequest, when receiving an InfoRequest the FDI Client does not wait for the user to acknowledge the message. Once the message has been displayed, the FDI Client immediately sends an InfoResponse back to the FDI Server.

The XML schema for an InfoRequestT type is:

```
<xs:complexType name="InfoRequestT">
   <xs:sequence>
      <xs:element name="Message" type="xs:string"/>
   </xs:sequence>
</xs:complexType>
```

The elements of an InfoRequestT type are described in Table A.28.

Table A.28 – Elements of InfoRequestT

Element	Description
Message	This required element specifies the message to be displayed to the user.

A.32 InputRequestT

This type specifies a request sent from an FDI Server to an FDI Client when the user needs to modify a value during an Action. Once the user has finished modifying the value, the FDI Client sends an InputResponse back to the FDI Server.

The XML schema for an InputRequestT type is:

```
<xs:complexType name="InputRequestT">
   <xs:sequence>
      <xs:element name="Prompt" type="xs:string"/>
      <xs:element name="InputValue" type="clnt:InputValueT"/>
   </xs:sequence>
</xs:complexType>
```

The elements of an InputRequestT type are described in Table A.29.

Table A.29 – Elements of InputRequestT

Element	Description
Prompt	This required element specifies the prompt to be displayed to the user.
InputValue	This required element specifies the value to be edited by the user.

A.33 InputResponseT

The response sent by an FDI Client to an FDI Server as a result of processing an InputRequest.

The XML schema for an InputResponseT type is:

```
<xs:complexType name="InputResponseT">
   <xs:sequence>
      <xs:element name="InputValue" type="clnt:VariantT"/>
   </xs:sequence>
</xs:complexType>
```

The elements of an InputResponseT type are described in Table A.30.

Table A.30 – Elements of InputResponseT

Element	Description
InputValue	This required element specifies the value the user specified.

A.34 InputValueT

This type specifies the value to be modified by the user during an InputRequest.

The XML schema for an InputValueT type is:

```
<xs:complexType name="InputValueT">
   <xs:complexContent>
      <xs:extension base="clnt:LabelHelpT">
         <xs:sequence>
            <xs:element name="InitialValue" type="clnt:VariantT"/>
            <xs:element name="Type" type="clnt:InputValueTypeT" minOccurs="0"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The elements of an InputValueT type are described in Table A.31.

Table A.31 – Elements of InputValueT

Element	Description
InitialValue	This required element specifies the initial value shown to the user.
Type	This optional element specifies the type of the value.

A.35 InputValueTypeT

This type specifies the type of the value modified by the user during an InputRequest.

The XML schema for an InputValueTypeT type is:

```
<xs:complexType name="InputValueTypeT">
   <xs:choice>
      <xs:element name="NumericType" type="clnt:NumericDataT"/>
      <xs:element name="StringType" type="clnt:StringT"/>
      <xs:element name="Enumeration" type="clnt:EnumerationItemListT">
         <xs:unique name="InputValueUniqueEnumValue">
            <xs:selector xpath="clnt:EnumerationItem/clnt:Value"/>
            <xs:field xpath="."/>
         </xs:unique>
      </xs:element>
      <xs:element name="BitEnumeration" type="clnt:BitEnumerationItemListT">
         <xs:unique name="InputValueUniqueBitMask">
            <xs:selector xpath="clnt:BitEnumerationItem/clnt:Value"/>
            <xs:field xpath="."/>
         </xs:unique>
      </xs:element>
      <xs:element name="DateTimeType" type="clnt:DateTimeDataT"/>
   </xs:choice>
</xs:complexType>
```

The elements of an InputValueTypeT type are described in Table A.32.

Table A.32 – Elements of InputValueTypeT

Element	Description
NumericType	This optional element specifies an integer or floating point value.
StringType	This optional element specifies a string value.
Enumeration	This element specifies an enumeration. Each enumeration item defines a description and help text for a parameter value.
BitEnumeration	This element specifies an enumeration. Each enumeration item defines a description and help text for a bitmask of the parameter value.
DateTimeType	This element specifies a date or time value.

A.36 LabelHelpT

This type specifies an extended layout object with an additional localized help text.

The XML schema for a LabelHelpT type is:

```
<xs:complexType name="LabelHelpT" abstract="true">
   <xs:complexContent>
      <xs:extension base="clnt:LabelT">
         <xs:sequence>
            <xs:element name="Help" type="xs:string" minOccurs="0"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The elements of a LabelHelpT type are described in Table A.33.

Table A.33 – Elements of LabelHelpT

Element	Description
Help	This optional element specifies a localized help text of the element.

A.37 LabelT

This type specifies the layout object with a localized description.

The XML schema for a LabelT type is:

```
<xs:complexType name="LabelT">
   <xs:sequence>
      <xs:element name="Label" type="xs:string"/>
   </xs:sequence>
</xs:complexType>
```

The elements of a LabelT type are described in Table A.34.

Table A.34 – Elements of LabelT

Element	Description
Label	This required element specifies the localized name of the element.

A.38 LineTypeT

This type specifies the type of line on a chart or a graph. All lines of the same type shall have the same visual appearance, for example all Data0 lines will appear the same and all Data1 lines will appear the same, but the two sets of lines shall be visually distinct.

The XML schema for a LineTypeT enumeration type is:

```
<xs:simpleType name="LineTypeT">
   <xs:restriction base="xs:string">
      <xs:enumeration value="Data"/>
      <xs:enumeration value="Data0"/>
      <xs:enumeration value="Data1"/>
      <xs:enumeration value="Data2"/>
      <xs:enumeration value="Data3"/>
      <xs:enumeration value="Data4"/>
      <xs:enumeration value="Data5"/>
      <xs:enumeration value="Data6"/>
      <xs:enumeration value="Data7"/>
      <xs:enumeration value="Data8"/>
      <xs:enumeration value="Data9"/>
      <xs:enumeration value="LowLimit"/>
      <xs:enumeration value="LowLowLimit"/>
      <xs:enumeration value="HighLimit"/>
      <xs:enumeration value="HighHighLimit"/>
      <xs:enumeration value="Transparent"/>
   </xs:restriction>
</xs:simpleType>
```

The enumeration values of a LineTypeT enumeration type are described in Table A.35.

Table A.35 – Enumerations of LineTypeT

Enumeration	Description
Data	The line represents data.
Data0	The line represents data.
Data1	The line represents data.
Data2	The line represents data.
Data3	The line represents data.
Data4	The line represents data.
Data5	The line represents data.
Data6	The line represents data.
Data7	The line represents data.
Data8	The line represents data.
Data9	The line represents data.
LowLimit	The line represents a low limit.
LowLowLimit	The line represents a low low limit.
HighLimit	The line represent a high limit.
HighHighLimit	The line represents a high high limit.
Transparent	Represents a transparent line type.

A.39 MenuT

This type specifies the layout of a Window, Dialog, Page, Group, Menu, or Table.

The XML schema for a MenuT type is:

```
<xs:complexType name="MenuT">
   <xs:complexContent>
      <xs:extension base="clnt:UiElementT">
         <xs:sequence>
            <xs:element name="InitActionList" type="clnt:ActionListT" minOccurs="0"/>
            <xs:element name="PreEditActionList" type="clnt:ActionListT"
               minOccurs="0"/>
            <xs:element name="PostEditActionList" type="clnt:ActionListT"
               minOccurs="0"/>
            <xs:element name="ExitActionList" type="clnt:ActionListT" minOccurs="0"/>
            <xs:element name="Access" type="clnt:AccessT" minOccurs="0"/>
            <xs:element name="ItemList">
               <xs:complexType>
```

```
                <xs:choice minOccurs="0" maxOccurs="unbounded">
                   <xs:element name="Menu" type="clnt:MenuReferenceT"/>
                   <xs:element name="Chart" type="clnt:ChartT"/>
                   <xs:element name="Graph" type="clnt:GraphT"/>
                   <xs:element name="Grid" type="clnt:GridT"/>
                   <xs:element name="Image" type="clnt:ImageT"/>
                   <xs:element name="Text" type="xs:string"/>
                   <xs:element name="Parameter" type="clnt:ParameterT"/>
                   <xs:element name="Plugin" type="clnt:PluginT"/>
                   <xs:element name="Action" type="clnt:ActionT"/>
                   <xs:element name="RowBreak" type="clnt:RowBreakT"/>
                   <xs:element name="ColumnBreak" type="clnt:ColumnBreakT"/>
                   <xs:element name="Separator" type="clnt:SeparatorT"/>
                </xs:choice>
             </xs:complexType>
          </xs:element>
       </xs:sequence>
       <xs:attribute name="Style" type="clnt:MenuStyleT" use="optional"/>
       <xs:attribute name="NodePath" type="xs:string" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The attributes of a MenuT type are described in Table A.36.

Table A.36 – Attributes of MenuT

Attribute	Description
Style	This optional attribute determines the menu style.
NodePath	This is a mandatory attribute which gives the qualified path of the Node in the Device Model.

The elements of a MenuT type are described in Table A.37.

Table A.37 – Elements of MenuT

Element	Description
InitActionList	This optional element specifies the Actions to be executed before the menu is activated.
PreEditActionList	This optional element specifies the Actions to be executed immediately after the element is activated.
PostEditActionList	This optional element specifies the Actions to be executed that after the user has finished processing the element.
ExitActionList	This optional element specifies the Actions to be executed when the menu is deactivated.
Access	This optional element specifies whether the menu shall be used ONLINE or OFFLINE
ItemList	This optional element specifies the base content of the menu.

A.40 MenuReferenceT

This type specifies the layout of a Window, Dialog, Page, Group, Menu, or Table.

The XML schema for a MenuReferenceT type is:

```
<xs:complexType name="MenuReferenceT">
   <xs:complexContent>
      <xs:extension base="clnt:UiElementT">
         <xs:sequence>
            <xs:element name="Review" type="xs:boolean" minOccurs="0"/>
         </xs:sequence>
         <xs:attribute name="Style" type="clnt:MenuStyleT" use="optional"/>
         <xs:attribute name="NodePath" type="xs:string" use="required"/>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The attributes of a MenuReferenceT type are described in Table A.38.

Table A.38 – Attributes of MenuReferenceT

Attribute	Description
Style	This optional attribute determines the menu style.
NodePath	This is mandatory attribute which gives the qualified path of the Node in the Device Model.

The elements of a MenuReferenceT type are described in Table A.39.

Table A.39 – Elements of MenuReferenceT

Element	Description
Review	This optional element specifies all referenced items of type VARIABLE appearing in the GUI or its subcomponents shall assume the qualifiers as READ ONLY.

A.41 MenuStyleT

This type specifies the style of a menu.

The XML schema for a MenuStyleT enumeration type is:

```
<xs:simpleType name="MenuStyleT">
   <xs:restriction base="xs:string">
      <xs:enumeration value="Menu"/>
      <xs:enumeration value="Group"/>
      <xs:enumeration value="Page"/>
      <xs:enumeration value="Table"/>
      <xs:enumeration value="Window"/>
      <xs:enumeration value="Dialog"/>
      <xs:enumeration value="Record"/>
      <xs:enumeration value="Collection"/>
      <xs:enumeration value="ItemArray"/>
      <xs:enumeration value="EditDisplay"/>
   </xs:restriction>
</xs:simpleType>
```

The enumeration values of a MenuStyleT enumeration type are described in Table A.40.

Table A.40 – Enumerations of MenuStyleT

Enumeration	Description
Menu	EDD Menu Style 'MENU'
Group	EDD Menu Style 'GROUP'
Page	EDD Menu Style 'PAGE'
Table	EDD Menu Style 'TABLE'
Window	EDD Menu Style 'WINDOW'
Dialog	EDD Menu Style 'DIALOG'
Record	Menu represents EDD 'RECORD'
Collection	Menu represents EDD 'COLLECTION'
ItemArray	Menu represents EDD 'ITEMARRAY'
EditDisplay	Menu represents EDD 'EDIT_DISPLAY'

A.42 NumericDataT

This type specifies the data type of an integer, boolean or floating point value.

The XML schema for a NumericDataT enumeration type is:

```
<xs:simpleType name="NumericDataT">
   <xs:restriction base="xs:string">
      <xs:enumeration value="SignedInteger"/>
      <xs:enumeration value="UnsignedInteger"/>
      <xs:enumeration value="FloatingPoint"/>
      <xs:enumeration value="Double"/>
      <xs:enumeration value="Boolean"/>
   </xs:restriction>
</xs:simpleType>
```

The enumeration values of a NumericDataT enumeration type are described in Table A.41.

Table A.41 – Enumerations of NumericDataT

Enumeration	Description
SignedInteger	A signed integer.
UnsignedInteger	An unsigned integer.
FloatingPoint	A floating point number.
Double	A Double point number.
Boolean	A boolean value.

A.43 NumericTemplateT

This type specifies template used to display a numeric parameter value.

The XML schema for a NumericTemplateT type is:

```
<xs:complexType name="NumericTemplateT">
```

```
  <xs:sequence>
    <xs:element name="DataType" type="clnt:NumericDataT"/>
    <xs:element name="Options" type="clnt:VariantOptionListT" minOccurs="0"
      maxOccurs="1">
      <xs:unique name="UiTemplateUniqueVariantValue">
        <xs:selector xpath="clnt:Option/clnt:Value/*"/>
        <xs:field xpath="."/>
      </xs:unique>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

The elements of a NumericTemplateT type are described in Table A.42.

Table A.42 – Elements of NumericTemplateT

Element	Description
DataType	This required element specifies the type of the numeric parameter.
Options	This optional element specifies the list of options the user may select for the string.

A.44 OptionListT

This type specifies a list of Option elements. The selection response contains the 0-based index of the selected option.

The XML schema for an OptionListT type is:

```
<xs:complexType name="OptionListT">
  <xs:sequence maxOccurs="unbounded">
    <xs:element name="Option" type="clnt:LabelT"/>
  </xs:sequence>
</xs:complexType>
```

The elements of an OptionListT type are described in Table A.43.

Table A.43 – Elements of OptionListT

Element	Description
Option	An element of the list.

A.45 OrientationT

This type specifies a direction.

The XML schema for an OrientationT enumeration type is:

```
<xs:simpleType name="OrientationT">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Horizontal"/>
    <xs:enumeration value="Vertical"/>
  </xs:restriction>
</xs:simpleType>
```

The enumeration values of an OrientationT enumeration type are described in Table A.44.

Table A.44 – Enumerations of OrientationT

Enumeration	Description
Horizontal	Left to right and right to left.
Vertical	Top to bottom and bottom to top.

A.46 ParameterInputRequestT

This type specifies a request sent from an FDI Server to an FDI Client when the user needs to modify a value during an Action. Once the user has finished modifying the value, the FDI Client sends an InputResponse back to the FDI Server.

The XML schema for a ParameterInputRequestT type is:

```
<xs:complexType name="ParameterInputRequestT">
   <xs:sequence>
      <xs:element name="Prompt" type="xs:string"/>
      <xs:element name="Parameter" type="clnt:ParameterT"/>
   </xs:sequence>
</xs:complexType>
```

The elements of a ParameterInputRequestT type are described in Table A.45.

Table A.45 – Elements of ParameterInputRequestT

Element	Description
Prompt	This required element specifies the prompt to be displayed to the user.
Parameter	This required element specifies the value to be edited by the user.

A.47 ParameterListT

This type specifies a list of Parameter elements.

The XML schema for a ParameterListT type is:

```
<xs:complexType name="ParameterListT">
   <xs:sequence maxOccurs="unbounded">
      <xs:element name="Parameter" type="clnt:ParameterT"/>
   </xs:sequence>
</xs:complexType>
```

The elements of a ParameterListT type are described in Table A.46.

Table A.46 – Elements of ParameterListT

Element	Description
Parameter	An element of the list.

A.48 ParameterT

This type specifies a device or block parameter that is displayed in a user interface element such as a Window or Dialog.

The XML schema for a ParameterT type is:

```
<xs:complexType name="ParameterT">
   <xs:complexContent>
      <xs:extension base="clnt:UiElementSizeableT">
         <xs:sequence>
            <xs:element name="UITemplate" type="clnt:UiTemplateT"/>
            <xs:element name="Unit" type="xs:string" minOccurs="0"/>
            <xs:element name="Handling" type="clnt:HandlingT" default="rw"
               minOccurs="0"/>
            <xs:element name="RangeList" type="clnt:RangeListT" minOccurs="0"/>
            <xs:element name="DisplayFormat" type="clnt:FormatSpecifierT"
               minOccurs="0"/>
            <xs:element name="EditFormat" type="clnt:FormatSpecifierT"
               minOccurs="0"/>
            <xs:element name="TimeFormat" type="xs:string" minOccurs="0"/>
            <xs:element name="TimeScale" type="clnt:TimeScaleT" minOccurs="0"/>
            <xs:element name="DisplayLabel" type="xs:boolean" default="true"
               minOccurs="0"/>
            <xs:element name="DisplayValue" type="xs:boolean" default="true"
               minOccurs="0"/>
            <xs:element name="DisplayUnit" type="xs:boolean" default="true"
               minOccurs="0"/>
            <xs:element name="PreEditActionList" type="clnt:ActionListT"
               minOccurs="0"/>
            <xs:element name="PostEditActionList" type="clnt:ActionListT"
               minOccurs="0"/>
            <xs:element name="ScalingFactor" type="xs:double" minOccurs="0"/>
            <xs:element name="Class" type="clnt:ParameterClassT" minOccurs="0"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The elements of a ParameterT type are described in Table A.47.

Table A.47 – Elements of ParameterT

Element	Description
UITemplate	This element specifies the complex UI template used to display a parameter. UI templates modify how parameter values are displayed. The default UI template displays a parameter value as plain text.
Unit	This optional element specifies the engineering unit of the parameter. The default is the parameter has no engineering unit.
Handling	This optional element specifies whether or not the parameter may be modified. The default is ReadWrite.
RangeList	This optional element specifies the minimum and maximum values to which the user may set the parameter.
DisplayFormat	This optional element specifies the format to be used when displaying the value of the parameter as specified in IEC 61804-3. The default is dependent upon the data type of the parameter and is determined by the FDI Client.
EditFormat	This optional element specifies the format to be used when modifying the value of the parameter as specified in IEC 61804-3. The default is dependent upon the data type of the parameter and is determined by the FDI Client.
TimeFormat	This optional element specifies the format to be used when displaying and modifying the value of parameters of date or time types.
TimeScale	This optional element specifies the TIME_SCALE of parameters of date or time types.
DisplayLabel	This optional element specifies whether or not the parameter's label should be displayed. The default is true.
DisplayValue	This optional element specifies whether or not the parameter's value should be displayed. The default is true.
DisplayUnit	This optional element specifies whether or not the parameter's unit should be displayed. The default is true.
PreEditActionList	This optional element specifies the Actions to be executed before the parameter is modified by the user.
PostEditActionList	This optional element specifies the Actions to be executed after the parameter is modified by the user.
ScalingFactor	This element specifies the scaling factor to be used when the value of the parameter will be shown to the user. This visible value is the result of the multiplication of the factor and the parameter value, returned by the device.
Class	This optional element specifies the EDD CLASS attribute of VARIABLE.

A.49 PluginT

This type specifies a User Interface Plug-in that is referenced from a user interface element such as a Window or Dialog.

The XML schema for a PluginT type is:

```
<xs:complexType name="PluginT">
   <xs:complexContent>
      <xs:extension base="clnt:UiElementT">
         <xs:sequence>
            <xs:element name="Identifier">
               <xs:simpleType>
                  <xs:restriction base="xs:string">
                  <xs:pattern value="[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-
                     [0-9a-fA-F]{4}-[0-9a-fA-F]{12}"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:element>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

The elements of a PluginT type are described in Table A.48.

Table A.48 – Elements of PluginT

Element	Description
Identifier	This required element specifies the Universally Unique Identifer (UUID) that identifies the plug-in.

A.50 RangeListT

This type specifies a list of Range elements.

The XML schema for a RangeListT type is:

```
<xs:complexType name="RangeListT">
   <xs:sequence maxOccurs="unbounded">
      <xs:element name="Range" type="clnt:RangeT"/>
   </xs:sequence>
</xs:complexType>
```

The elements of a RangeListT type are described in Table A.49.

Table A.49 – Elements of RangeListT

Element	Description
Range	An element of the list.

A.51 RangeT

This type specifies a range defined by a minimum and a maximum value.

The XML schema for a RangeT type is:

```
<xs:complexType name="RangeT">
   <xs:sequence>
      <xs:element name="MinimumValue" type="clnt:VariantT"/>
      <xs:element name="MaximumValue" type="clnt:VariantT" minOccurs="0"/>
   </xs:sequence>
</xs:complexType>
```

The elements of a RangeT type are described in Table A.50.

Table A.50 – Elements of RangeT

Element	Description
MinimumValue	This required element specifies the minimum value of the range.
MaximumValue	This required element specifies the maximum value of the range.

A.52 ResponseT

This type specifies the response sent by an FDI Client to an FDI Server as a result of processing an Action request.

The XML schema for a ResponseT type is:

```
<xs:complexType name="ResponseT"/>
```

A.53 RowBreakT

This type specifies a row break.

The XML schema for a RowBreakT type is:

```
<xs:complexType name="RowBreakT"/>
```

A.54 ScalingT

This type specifies a method of scaling values from a lower bound to an upper bound.

The XML schema for a ScalingT enumeration type is:

```
<xs:simpleType name="ScalingT">
   <xs:restriction base="xs:string">
      <xs:enumeration value="Linear"/>
      <xs:enumeration value="Logarithmic"/>
   </xs:restriction>
</xs:simpleType>
```

The enumeration values of a ScalingT enumeration type are described in Table A.51.

Table A.51 – Enumerations of ScalingT

Enumeration	Description
Linear	The values are scaled linearly.
Logarithmic	The values are scaled logarithmically via the natural logarithm.

A.55 SelectionRequestT

This type specifies a request sent from an FDI Server to an FDI Client when the user needs to choose from a list of options during an Action. Once the user has selected one of the options, the FDI Client sends an SelectionResponse back to the FDI Server.

The XML schema for a SelectionRequestT type is:

```
<xs:complexType name="SelectionRequestT">
   <xs:sequence>
      <xs:element name="Prompt" type="xs:string"/>
      <xs:element name="OptionList" type="clnt:OptionListT">
         <xs:unique name="UniqueOptionIdentifier">
            <xs:selector xpath="clnt:Option/clnt:Label"/>
            <xs:field xpath="."/>
         </xs:unique>
      </xs:element>
   </xs:sequence>
</xs:complexType>
```

The elements of a SelectionRequestT type are described in Table A.52.

Table A.52 – Elements of SelectionRequestT

Element	Description
Prompt	This required element specifies the prompt to be displayed to the user.
OptionList	This required element specifies the list of options the user may select from.

A.56 SelectionResponseT

This type specifies the response sent by an FDI Client to an FDI Server as a result of processing an SelectionRequest.

The XML schema for a SelectionResponseT type is:

```
<xs:complexType name="SelectionResponseT">
   <xs:sequence>
      <xs:element name="SelectedOption" type="xs:unsignedLong"/>
   </xs:sequence>
</xs:complexType>
```

The elements of a SelectionResponseT type are described in Table A.53.

Table A.53 – Elements of SelectionResponseT

Element	Description
SelectedOption	The 0-based index of the selected option.

A.57 SeparatorT

This type specifies a separator.

The XML schema for a SeparatorT type is:

```
<xs:complexType name="SeparatorT"/>
```

A.58 SizeT

This type specifies the relative size of an item.

The XML schema for a SizeT enumeration type is:

```
<xs:simpleType name="SizeT">
   <xs:restriction base="xs:string">
      <xs:enumeration value="XXX_SMALL"/>
      <xs:enumeration value="XX_SMALL"/>
      <xs:enumeration value="X_SMALL"/>
      <xs:enumeration value="SMALL"/>
      <xs:enumeration value="MEDIUM"/>
      <xs:enumeration value="LARGE"/>
      <xs:enumeration value="X_LARGE"/>
      <xs:enumeration value="XX_LARGE"/>
   </xs:restriction>
</xs:simpleType>
```

The enumeration values of a SizeT enumeration type are described in Table A.54.

Table A.54 – Enumerations of SizeT

Enumeration	Description
XXX_SMALL	Extra extra extra small.
XX_SMALL	Extra extra small.
X_SMALL	Extra small.
SMALL	Small.
MEDIUM	Medium.
LARGE	Large.
X_LARGE	Extra large.
XX_LARGE	Extra extra large.

A.59 ParameterClassT

This type specifies the EDD CLASS type definition for VARIABLE.

The XML schema for a ParameterClassT enumeration type is:

```
<xs:simpleType name="ParameterClassT">
   <xs:list>
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="ALARM"/>
            <xs:enumeration value="ANALOG_INPUT"/>
            <xs:enumeration value="ANALOG_OUTPUT"/>
            <xs:enumeration value="COMPUTATION"/>
            <xs:enumeration value="CONTAINED"/>
            <xs:enumeration value="CORRECTION"/>
            <xs:enumeration value="DEVICE"/>
            <xs:enumeration value="SPECIALIST"/>
            <xs:enumeration value="DIAGNOSTIC"/>
            <xs:enumeration value="DIGITAL_INPUT"/>
            <xs:enumeration value="DIGITAL_OUTPUT"/>
```

```
          <xs:enumeration value="DISCRETE_INPUT"/>
          <xs:enumeration value="DISCRETE_OUTPUT"/>
          <xs:enumeration value="DYNAMIC"/>
          <xs:enumeration value="FREQUENCY_INPUT"/>
          <xs:enumeration value="FREQUENCY_OUTPUT"/>
          <xs:enumeration value="HART"/>
          <xs:enumeration value="INPUT"/>
          <xs:enumeration value="IS_CONFIG"/>
          <xs:enumeration value="LOCAL"/>
          <xs:enumeration value="LOCAL_DISPLAY"/>
          <xs:enumeration value="OPERATE"/>
          <xs:enumeration value="OPTIONAL"/>
          <xs:enumeration value="OUTPUT"/>
          <xs:enumeration value="SERVICE"/>
          <xs:enumeration value="TEMPORARY"/>
          <xs:enumeration value="TUNE"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:list>
</xs:simpleType>
```

The enumeration values of a ParameterClassT enumeration type are described in Table A.55.

Table A.55 – Enumerations of ParameterClassT

Enumeration	Description
ALARM	The VARIABLE contains alarm limits.
ANALOG_INPUT	The VARIABLE is part of an analog input block.
ANALOG_OUTPUT	The VARIABLE is part of an analog output block.
COMPUTATION	The VARIABLE is part of a computation block.
CONTAINED	The VARIABLE represents the physical characteristics of the device.
CORRECTION	The VARIABLE is part of the correction block.
DEVICE	The VARIABLE represents the physical characteristics of the device.
SPECIALIST	The VARIABLE is a configuration parameter. Permission to modify this VARIABLE may only be granted to users that are specialists for this device.
DIAGNOSTIC	The VARIABLE indicates the device status.
DIGITAL_INPUT	The VARIABLE is part of a digital input block.
DIGITAL_OUTPUT	The VARIABLE is part of a digital output block.
DISCRETE_INPUT	The VARIABLE is part of a discrete input block.
DISCRETE_OUTPUT	The VARIABLE is part of a discrete output block.
DYNAMIC	The VARIABLE is modified by the device without stimulus from the network.
FREQUENCY_INPUT	The VARIABLE is part of a frequency input block.
FREQUENCY_OUTPUT	The VARIABLE is part of a frequency output block.
HART	The VARIABLE is part of the HART block which characterizes the HART interface.
INPUT	The VARIABLE is part of an input block.
IS_CONFIG	A modification of the VARIABLE results in setting the revision counter or configuration-changed bit, as applicable.
LOCAL	The VARIABLE is locally used by the EDD application.
LOCAL_DISPLAY	The VARIABLE is part of the local display block.
OPERATE	The VARIABLE is used to control a block’s operation.

Enumeration	Description
OPTIONAL	The VARIABLE may not be present in the device.
OUTPUT	The VARIABLE is part of the output block.
SERVICE	The VARIABLE is used when performing routine maintenance.
TEMPORARY	The VARIABLE is initialized to its DEFAULT_VALUE in each EDD application session. LOCAL variables may be persisted by the EDD application; TEMPORARY variables are never persisted.
TUNE	The VARIABLE is used to tune the algorithm of a block.

A.60 ActionClassT

This type specifies the EDD CLASS definition of an Action

The XML schema for a ActionClassT enumeration type is:

```
<xs:simpleType name="ActionClassT">
   <xs:list>
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="ALARM"/>
            <xs:enumeration value="ANALOG_OUTPUT"/>
            <xs:enumeration value="COMPUTATION"/>
            <xs:enumeration value="CONTAINED"/>
            <xs:enumeration value="CORRECTION"/>
            <xs:enumeration value="DEVICE"/>
            <xs:enumeration value="SPECIALIST"/>
            <xs:enumeration value="DIAGNOSTIC"/>
            <xs:enumeration value="DISCRETE"/>
            <xs:enumeration value="FREQUENCY"/>
            <xs:enumeration value="HART"/>
            <xs:enumeration value="INPUT"/>
            <xs:enumeration value="LOCAL_DISPLAY"/>
            <xs:enumeration value="OPERATE"/>
            <xs:enumeration value="OUTPUT"/>
            <xs:enumeration value="SERVICE"/>
            <xs:enumeration value="TUNE"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:list>
</xs:simpleType>
```

The enumeration values of a ActionClassT enumeration type are described in Table A.56.

Table A.56 – Enumerations of ActionClassT

Enumeration	Description
ALARM	The METHOD is associated with an alarm (e.g., specifying alarm limits, indicating alarm status, etc.)
ANALOG_OUTPUT	The METHOD is part of an analog output block.
COMPUTATION	The METHOD is part of a computation block.
CONTAINED	The METHOD represents the physical characteristics of the device.
CORRECTION	The METHOD supports the transducers in the field device. The METHOD may establish transducer limits, provide signal damping, indicate the transducer's value, linearize or calibrate the transducer, etc.
DEVICE	The METHOD specifies the physical characteristics of the device.
SPECIALIST	Permission to execute this METHOD may only be granted to users that are specialists for this device.
DIAGNOSTIC	The METHOD evaluates the device status.
DISCRETE	The METHOD supports the discrete and or digital I/O of the device.
FREQUENCY	The METHOD supports the frequency I/O of the device.
HART	The METHOD is part of the HART block which characterizes the HART interface. There is only one HART block.
INPUT	The METHOD is part of an input block. An input block is a special kind of computation block which does unit conversions, scaling, and damping. The input block parameters can be determined by the output of another block.
LOCAL_DISPLAY	The METHOD is part of the local display block. A local display block contains the VARIABLEs associated with the local interface (keyboard, display, etc.) of the device.
OPERATE	The METHOD is used to control a block’s operation.
OUTPUT	The METHOD is part of the output block. The values of output parameters may be accessed by another block input.
SERVICE	The METHOD is used when performing routine maintenance.
TUNE	The METHOD is used to tune the algorithm of a block.

A.61 SourceListT

This type specifies a list of Source elements.

The XML schema for a SourceListT type is:

```
<xs:complexType name="SourceListT">
   <xs:sequence maxOccurs="unbounded">
      <xs:element name="Source" type="clnt:SourceT"/>
   </xs:sequence>
</xs:complexType>
```

The elements of a SourceListT type are described in Table A.57.

Table A.57 – Elements of SourceListT

Element	Description
Source	An element of the list.

A.62 SourceT

This type specifies a source of data values displayed by a chart.

The XML schema for a SourceT type is:

```
<xs:complexType name="SourceT">
   <xs:complexContent>
      <xs:extension base="clnt:DiagramLineT">
         <xs:sequence>
            <xs:element name="ParameterList" type="clnt:ParameterListT"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The elements of a SourceT type are described in Table A.58.

Table A.58 – Elements of SourceT

Element	Description
ParameterList	This required element specifies the parameters to be sampled and displayed.

A.63 StringDataT

This type specifies that data type of a string value.

The XML schema for a StringDataT enumeration type is:

```
<xs:simpleType name="StringDataT">
   <xs:restriction base="xs:string">
      <xs:enumeration value="ASCII"/>
      <xs:enumeration value="BITSTRING"/>
      <xs:enumeration value="EUC"/>
      <xs:enumeration value="PACKED_ASCII"/>
      <xs:enumeration value="PASSWORD"/>
      <xs:enumeration value="VISIBLE"/>
      <xs:enumeration value="OCTET"/>
   </xs:restriction>
</xs:simpleType>
```

The enumeration values of a StringDataT enumeration type are described in Table A.59.

Table A.59 – Enumerations of StringDataT

Enumeration	Description
ASCII	String of ASCII characters.
BITSTRING	Variables of data type BIT_ENUMERATED are string constants which identify the position of a character of the VARIABLE value with its position.
EUC	(Extended Unit code) – is the internal code for specific characters (for example, China: EUC-CN, Taiwan EUC-TW). EUC is used to handle East Asian languages (ISO/IEC 10646-1).
PACKED_ASCII	Is a subset of ASCII produced by removing the two most significant bits of each ASCII character.
PASSWORD	Is a string type and is intended for specifying password strings. Except for how the variable is presented to the user, password and ASCII string types are identical.
VISIBLE	This string is an ordered sequence of characters from the ISO/IEC 10646-1 and ISO/IEC 2375 character set.
OCTET	Is for specifying a sequence of unformatted binary data whose definition is determined by the implementation of the device.

A.64 StringTemplateT

This type specifies the template used to display a string parameter value.

The XML schema for a StringTemplateT type is:

```
<xs:complexType name="StringTemplateT">
   <xs:sequence>
      <xs:element name="DataType" type="clnt:StringDataT"/>
      <xs:element name="Options" type="clnt:StringOptionListT" minOccurs="0"
         maxOccurs="1">
         <xs:unique name="UiTemplateUniqueStringValue">
            <xs:selector xpath="clnt:Option/clnt:Value"/>
            <xs:field xpath="."/>
         </xs:unique>
      </xs:element>
   </xs:sequence>
</xs:complexType>
```

The elements of a StringTemplateT type are described in Table A.60.

Table A.60 – Elements of StringTemplateT

Element	Description
DataType	This required element specifies the type of the string parameter.
Options	This optional element specifies the list of options the user may select for the string.

A.65 StringOptionListT

This type specifies the list of options to choose for the string parameter type.

The XML schema for a StringOptionListT type is:

```
<xs:complexType name="StringOptionListT">
   <xs:sequence maxOccurs="unbounded">
      <xs:element name="Option" type="clnt:StringOptionT"/>
   </xs:sequence>
</xs:complexType>
```

The elements of a StringOptionListT type are described in Table A.61.

Table A.61 – Elements of StringOptionListT

Element	Description
Option	An element of the string options.

A.66 StringOptionT

This type specifies one of the possible values of string options.

The XML schema for a StringOptionT type is:

```
<xs:complexType name="StringOptionT">
   <xs:complexContent>
      <xs:extension base="clnt:LabelHelpT">
         <xs:sequence>
            <xs:element name="Value" type="xs:string"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The elements of a StringOptionT type are described in Table A.62.

Table A.62 – Elements of StringOptionT

Element	Description
Value	This required element specifies the unique identifier of the list entry in the context of the parent list element.

A.67 StringT

This type specifies the type of a string value.

The XML schema for a StringT type is:

```
<xs:complexType name="StringT">
   <xs:sequence>
      <xs:element name="DataType" type="clnt:StringDataT"/>
      <xs:element name="Length" type="xs:unsignedInt"/>
   </xs:sequence>
</xs:complexType>
```

The elements of a StringT type are described in Table A.63.

Table A.63 – Elements of StringT

Element	Description
DataType	This required element specifies the data type of the string.
Length	This required element specifies the length of the string, in characters not in octets.

A.68 TimeScaleT

This enum type specifies the options for the EDD timescale modifier.

The XML schema for a TimeScaleT enumeration type is:

```
<xs:simpleType name="TimeScaleT">
   <xs:restriction base="xs:string">
      <xs:enumeration value="SECONDS"/>
      <xs:enumeration value="MINUTES"/>
      <xs:enumeration value="HOURS"/>
   </xs:restriction>
</xs:simpleType>
```

The enumeration values of a TimeScaleT enumeration type are described in Table A.64.

Table A.64 – Enumerations of TimeScaleT

Enumeration	Description
SECONDS	The access shall be done ONLINE.
MINUTES	The access shall be done OFFLINE.
HOURS	The access shall be done OFFLINE.

A.69 UidLayoutInformation

This element is the root entry element containing the UID.

The XML schema for a UidLayoutInformation element is:

```
<xs:element name="UidLayoutInformation">
   <xs:complexType>
      <xs:sequence>
         <xs:choice>
            <xs:element name="Menu" type="clnt:MenuT"/>
            <xs:element name="ActionRequest" type="clnt:ActionRequestT"/>
            <xs:element name="ActionResponse" type="clnt:ActionResponseT"/>
         </xs:choice>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

The elements of a UidLayoutInformation element are described in Table A.65.

Table A.65 – Elements of UidLayoutInformation

Element	Description
Menu	This optional element specifies a request from an FDI Server to an FDI Client to display a menu.
ActionRequest	This optional element specifies a request from an FDI Server to an FDI Client for the user to perform a specific action.
ActionResponse	This optional element specifies a request from an FDI Server to an FDI Client for a response to be displayed to the user.

A.70 UidRequestT

This type specifies a request sent from an FDI Server to an FDI Client when an advanced user interface needs to be shown to the user during an Action. The structure of the user interface is specified as a Dialog, Table, or Window. When the user interface is no longer needed, the FDI Client sends a UIDResponse back to the FDI Server.

The XML schema for a UidRequestT type is:

```
<xs:complexType name="UidRequestT">
   <xs:sequence>
      <xs:element name="UidRef" type="xs:string"/>
      <xs:element name="ButtonList" type="clnt:ButtonListT">
         <xs:unique name="UniqueButtonIdentifier">
            <xs:selector xpath="clnt:Button/clnt:Label"/>
            <xs:field xpath="."/>
         </xs:unique>
      </xs:element>
   </xs:sequence>
</xs:complexType>
```

The elements of a UidRequestT type are described in Table A.66.

Table A.66 – Elements of UidRequestT

Element	Description
UidRef	This required element specifies the Node path of the UID containing the definition of the Dialog, Table, or Window to be displayed.
ButtonList	This required element specifies the buttons displayed by the FDI Client that the user may press to indicate the user interface is no longer needed.

A.71 UidResponseT

This type specifies the response sent by an FDI Client to an FDI Server as a result of processing a SelectionRequest.

The XML schema for a UidResponseT type is:

```
<xs:complexType name="UidResponseT">
   <xs:sequence>
      <xs:element name="SelectedButton" type="xs:unsignedLong"/>
   </xs:sequence>
```

```
</xs:complexType>
```

The elements of a UidResponseT type are described in Table A.67.

Table A.67 – Elements of UidResponseT

Element	Description
SelectedButton	The 0-based index of the selected button. If the button list is not provided in the UID request, Next and Cancel buttons are shown. SelectedButton=0 means that the Next button was pressed.

A.72 UiElementSizeableT

The XML schema for a UiElementSizeableT type is:

```
<xs:complexType name="UiElementSizeableT">
   <xs:complexContent>
      <xs:extension base="clnt:UiElementT">
         <xs:sequence>
            <xs:element name="Height" type="clnt:SizeT" minOccurs="0"/>
            <xs:element name="Width" type="clnt:SizeT" minOccurs="0"/>
         </xs:sequence>
         <xs:attribute name="NodePath" type="xs:string" use="required"/>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The attributes of a UiElementSizeableT type are described in Table A.68.

Table A.68 – Attributes of UiElementSizeableT

Attribute	Description
NodePath	This is a mandatory attribute which specifies the qualified path of the Node in the Device Model. The path description helps in finding a Node in the Information Model.

The elements of a UiElementSizeableT type are described in Table A.69.

Table A.69 – Elements of UiElementSizeableT

Element	Description
Height	This optional element specifies the relative height of the element.
Width	This optional element specifies the relative width of the element.

A.73 UiElementT

This type specifies an extended layout object with a settable visibility.

The XML schema for a UiElementT type is:

```
<xs:complexType name="UiElementT">
   <xs:complexContent>
      <xs:extension base="clnt:LabelHelpT">
```

```
        <xs:sequence>
          <xs:element name="Visibility" type="xs:boolean" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The elements of a UiElementT type are described in Table A.70.

Table A.70 – Elements of UiElementT

Element	Description
Visibility	This optional element specifies whether the element is visible or not. The default is visible.

A.74 UiTemplateT

This type specifies templates used to display a parameter value. UI templates modify how parameter values are displayed.

The XML schema for a UiTemplateT type is:

```
<xs:complexType name="UiTemplateT">
  <xs:sequence>
    <xs:choice>
      <xs:element name="Enumeration" type="clnt:EnumerationItemListT">
        <xs:unique name="UiTemplateUniqueEnumValue">
          <xs:selector xpath="clnt:EnumerationItem/clnt:Value"/>
          <xs:field xpath="."/>
        </xs:unique>
      </xs:element>
      <xs:element name="BitEnumeration" type="clnt:BitEnumerationItemListT">
        <xs:unique name="UiTemplateUniqueBitMask">
          <xs:selector xpath="clnt:BitEnumerationItem/clnt:Value"/>
          <xs:field xpath="."/>
        </xs:unique>
      </xs:element>
      <xs:element name="String" type="clnt:StringTemplateT"/>
      <xs:element name="Arithmetic" type="clnt:NumericTemplateT"/>
      <xs:element name="DateTime" type="clnt:DateTimeDataT"/>
    </xs:choice>
    <xs:element name="UiTemplateSize" type="xs:int" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

The elements of a UiTemplateT type are described in Table A.71.

Table A.71 – Elements of UiTemplateT

Element	Description
Enumeration	This element specifies that the value shall be displayed with an UI template optimized for enumerations. Each enumeration item defines a description and help text for a parameter value.
BitEnumeration	This element specifies that the value shall be displayed with an UI template optimized for bit enumerations. Each enumeration item defines a description and help text for a bitmask of the parameter value.
String	This element specifies that the value shall be displayed with an UI template optimized for all values of String Types.
Arithmetic	This element specifies that the value shall be displayed with an UI template optimized for all values of Numeric Types.
DateTime	This element specifies that the value shall be displayed with an UI template optimized for all values of Date and Time Types.
UiTemplateSize	This is an optional element used to specify the size of the Arithmetic data types, for example Byte information for Signed and Unsigned integer and to specify the number of characters in case of string datatypes.

A.75 VariantT

This type specifies a value. The value is in raw format and not scaled.

The XML schema for a VariantT type is:

```
<xs:complexType name="VariantT">
   <xs:choice>
      <xs:element name="Float" type="xs:float"/>
      <xs:element name="Double" type="xs:double"/>
      <xs:element name="Integer" type="xs:integer"/>
      <xs:element name="UnsignedInteger" type="xs:nonNegativeInteger"/>
      <xs:element name="Date" type="xs:date"/>
      <xs:element name="DateTime" type="xs:dateTime"/>
      <xs:element name="Time" type="xs:time"/>
      <xs:element name="Duration" type="xs:duration"/>
      <xs:element name="String" type="xs:string"/>
      <xs:element name="Boolean" type="xs:boolean"/>
   </xs:choice>
</xs:complexType>
```

The elements of a VariantT type are described in Table A.72.

Table A.72 – Elements of VariantT

Element	Description
Float	This optional element specifies a single precision floating point value.
Double	This optional element specifies a double precision floating point value.
Integer	This optional element specifies a signed integer value.
UnsignedInteger	This optional element specifies an unsigned integer value.
Date	This optional element specifies a date value.
DateTime	This optional element specifies a date and time value.
Time	This optional element specifies a time value.
Duration	This optional element specifies a length of time.
String	This optional element specifies a string.
Boolean	This optional element specifies a boolean.

A.76 VariantOptionListT

This type specifies the list of options to choose for the numeric parameter type.

The XML schema for a VariantOptionListT type is:

```
<xs:complexType name="VariantOptionListT">
   <xs:sequence maxOccurs="unbounded">
      <xs:element name="Option" type="clnt:VariantOptionT"/>
   </xs:sequence>
</xs:complexType>
```

The elements of a VariantOptionListT type are described in Table A.73.

Table A.73 – Elements of VariantOptionListT

Element	Description
Option	An element of the numeric options.

A.77 VariantOptionT

This type specifies one of the possible values of numeric options.

The XML schema for a VariantOptionT type is:

```
<xs:complexType name="VariantOptionT">
   <xs:complexContent>
      <xs:extension base="clnt:LabelHelpT">
         <xs:sequence>
            <xs:element name="Value" type="clnt:VariantT"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The elements of a VariantOptionT type are described in Table A.74.

Table A.74 – Elements of VariantOptionT

Element	Description
Value	This required element specifies the unique identifier of the list entry in the context of the parent list element.

A.78 VectorListT

This type specifies a list of Vector elements.

The XML schema for a VectorListT type is:

```
<xs:complexType name="VectorListT">
   <xs:sequence maxOccurs="unbounded">
      <xs:element name="Vector" type="clnt:VectorT"/>
   </xs:sequence>
</xs:complexType>
```

The elements of a VectorListT type are described in Table A.75.

Table A.75 – Elements of VectorListT

Element	Description
Vector	An element of the list.

A.79 VectorT

This type specifies the content of a row or column in a grid.

The XML schema for a VectorT type is:

```
<xs:complexType name="VectorT">
   <xs:sequence>
      <xs:element name="Heading" type="xs:string"/>
      <xs:element name="Items">
         <xs:complexType>
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
               <xs:choice>
                  <xs:element name="Parameter" type="clnt:ParameterT"/>
                  <xs:element name="Value" type="clnt:VariantT"/>
               </xs:choice>
            </xs:sequence>
         </xs:complexType>
      </xs:element>
   </xs:sequence>
</xs:complexType>
```

The elements of a VectorT type are described in Table A.76.

Table A.76 – Elements of VectorT

Element	Description
Heading	This required element specifies the localized heading displayed along with the data.
Items	This required element specifies the data to be displayed, which may be parameters or static scalar values.

A.80 WaveformListT

This type specifies a list of Waveform elements.

The XML schema for a WaveformListT type is:

```
<xs:complexType name="WaveformListT">
   <xs:sequence maxOccurs="unbounded">
      <xs:element name="Waveform" type="clnt:WaveformT"/>
   </xs:sequence>
</xs:complexType>
```

The elements of a WaveformListT type are described in Table A.77.

Table A.77 – Elements of WaveformListT

Element	Description
Waveform	An element of the list.

A.81 WaveformT

This type specifies a single data set displayed on a graph. A graph may contain one or more waveforms.

The XML schema for a WaveformT type is:

```
<xs:complexType name="WaveformT">
   <xs:complexContent>
      <xs:extension base="clnt:DiagramLineT">
         <xs:sequence>
            <xs:element name="Handling" type="clnt:HandlingT" default="rw"
               minOccurs="0" maxOccurs="1"/>
            <xs:element name="WaveformType" type="clnt:WaveformTypeT" minOccurs="1"
               maxOccurs="1"/>
            <xs:element name="KeyPointList" type="clnt:WaveformKeyPointListT"
               minOccurs="0" maxOccurs="1"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The elements of a WaveformT type are described in Table A.78.

Table A.78 – Elements of WaveformT

Element	Description
Handling	This optional element specifies whether or not the waveform may be modified. The default is ReadWrite.
WaveformType	
KeyPointList	This optional element specifies a collection of key points that are displayed along with the waveform. The key points may or may not be points on the waveform itself.

A.82 WaveformTypeT

This is the abstract base type of all Waveform Types.

The XML schema for a WaveformTypeT type is:

```
<xs:complexType name="WaveformTypeT" abstract="true"/>
```

A.83 WaveformTypeHorizontalT

From EDD-Spec: The HORIZONTAL attribute specifies a WAVEFORM that contains horizontal lines.

The XML schema for a WaveformTypeHorizontalT type is:

```
<xs:complexType name="WaveformTypeHorizontalT">
   <xs:complexContent>
      <xs:extension base="clnt:WaveformTypeT">
         <xs:sequence>
            <xs:element name="Yvalues" type="clnt:WaveformVectorT" minOccurs="1"
               maxOccurs="1"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The elements of a WaveformTypeHorizontalT type are described in Table A.79.

Table A.79 – Elements of WaveformTypeHorizontalT

Element	Description
Yvalues	From EDD-Spec: The Y_VALUES attribute specifies the Y coordinate of each horizontal line in the WAVEFORM.

A.84 WaveformTypeVerticalT

From EDD-Spec: The VERTICAL attribute specifies a WAVEFORM that contains vertical lines.

The XML schema for a WaveformTypeVerticalT type is:

```
<xs:complexType name="WaveformTypeVerticalT">
   <xs:complexContent>
      <xs:extension base="clnt:WaveformTypeT">
         <xs:sequence>
            <xs:element name="Xvalues" type="clnt:WaveformVectorT" minOccurs="1"
               maxOccurs="1"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The elements of a WaveformTypeVerticalT type are described in Table A.80.

Table A.80 – Elements of WaveformTypeVerticalT

Element	Description
Xvalues	From EDD-Spec: The X_VALUES attribute specifies the X coordinate of each vertical line in the WAVEFORM.

A.85 WaveformTypeYTT

From EDD-Spec: The YT attribute specifies a WAVEFORM that contains a list of points that are defined via an initial X coordinate, an X increment between successive points and a list of Y values.

The XML schema for a WaveformTypeYTT type is:

```
<xs:complexType name="WaveformTypeYTT">
   <xs:complexContent>
      <xs:extension base="clnt:WaveformTypeT">
         <xs:sequence>
            <xs:element name="Yvalues" type="clnt:WaveformVectorT" minOccurs="1"
               maxOccurs="1"/>
            <xs:element name="Xinitial" type="clnt:VariantT" minOccurs="1"
               maxOccurs="1"/>
            <xs:element name="Xincrement" type="clnt:VariantT" minOccurs="1"
               maxOccurs="1"/>
            <xs:element name="NumberOfPoints" type="xs:nonNegativeInteger"
               minOccurs="0" maxOccurs="1"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The elements of a WaveformTypeYTT type are described in Table A.81.

Table A.81 – Elements of WaveformTypeYTT

Element	Description
Yvalues	From EDD-Spec:The Y_VALUES attribute specifies the Y coordinate of each point in the WAVEFORM.
Xinitial	From EDD-Spec:The X_INITIAL attribute specifies the X coordinate of the first point in the WAVEFORM.
Xincrement	From EDD-Spec:The X_INCREMENT attribute specifies difference between the X coordinates of adjacent points in the WAVEFORM.
NumberOfPoints	From EDD-Spec: The NUMBER_OF_POINTS attribute specifies the number of valid data points in X_VALUES. By default, the number of points in the WAVEFORM without a NUMBER_OF_POINTS attribute equals the size of X_VALUES.

A.86 WaveformTypeXYT

From EDD-Spec: The XY attribute specifies a WAVEFORM that contains a list of (x,y) points.

The XML schema for a WaveformTypeXYT type is:

```
<xs:complexType name="WaveformTypeXYT">
   <xs:complexContent>
      <xs:extension base="clnt:WaveformTypeT">
         <xs:sequence>
            <xs:element name="Xvalues" type="clnt:WaveformVectorT" minOccurs="1"
               maxOccurs="1"/>
            <xs:element name="Yvalues" type="clnt:WaveformVectorT" minOccurs="1"
               maxOccurs="1"/>
            <xs:element name="NumberOfPoints" type="xs:nonNegativeInteger"
               minOccurs="0" maxOccurs="1"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The elements of a WaveformTypeXYT type are described in Table A.82.

Table A.82 – Elements of WaveformTypeXYT

Element	Description
Xvalues	From EDD-Spec: The X_VALUES attribute specifies the X coordinate of each point in the WAVEFORM. For each X coordinate specified in X_VALUES there shall be a corresponding Y coordinate specified in Y_VALUES.
Yvalues	From EDD-Spec: The Y_VALUES attribute specifies the Y coordinate of each point in the WAVEFORM. For each Y coordinate specified in Y_VALUES there shall be a corresponding X coordinate specified in X_VALUES.
NumberOfPoints	From EDD-Spec: The NUMBER_OF_POINTS attribute specifies the number of valid data points in X_VALUES and Y_VALUES. By default, the number of points in the WAVEFORM without a NUMBER_OF_POINTS attribute equals the size of X_VALUES and Y_VALUES.

A.87 WaveformKeyPointListT

From EDD-Spec: The KEY_POINTS attribute specifies key points in the WAVEFORM that should be highlighted by the EDD application. The key points need not directly correspond to the data points specified via the TYPE attribute. The way in which these points are highlighted is defined by the EDD specification. By default, a WAVEFORM without a KEY_POINTS attribute should not have any of its points highlighted.

The XML schema for a WaveformKeyPointListT type is:

```
<xs:complexType name="WaveformKeyPointListT">
   <xs:complexContent>
      <xs:extension base="clnt:WaveformTypeT">
         <xs:sequence>
            <xs:element name="Xvalues" type="clnt:WaveformVectorT" minOccurs="1"
               maxOccurs="1"/>
            <xs:element name="Yvalues" type="clnt:WaveformVectorT" minOccurs="1"
               maxOccurs="1"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

The elements of a WaveformKeyPointListT type are described in Table A.83.

Table A.83 – Elements of WaveformKeyPointListT

Element	Description
Xvalues	From EDD-Spec: The X_VALUES attribute specifies the X coordinate of each point that is to be highlighted. For each X coordinate specified in X_VALUES there shall be a corresponding Y coordinate specified in Y_VALUES.
Yvalues	From EDD-Spec: The Y_VALUES attribute specifies the Y coordinate of each point that is to be highlighted. For each Y coordinate specified in Y_VALUES, there shall be a corresponding X coordinate specified in X_VALUES.

A.88 WaveformVectorT

WaveformVectorT represents the whole information of a waveform vector, for example the nodepath for the array of scalar values and additional information about these values such as Handling, Range, Type.

The XML schema for a WaveformVectorT type is:

```
<xs:complexType name="WaveformVectorT">
   <xs:sequence>
      <xs:element name="WaveformVectorElementList"
         type="clnt:WaveformVectorElementListT" minOccurs="1" maxOccurs="1"/>
   </xs:sequence>
   <xs:attribute name="DataArrayNodePath" type="xs:string" use="required"/>
</xs:complexType>
```

The attributes of a WaveformVectorT type are described in Table A.84.

Table A.84 – Attributes of WaveformVectorT

Attribute	Description
DataArrayNodePath	The nodepath that is used by the client to request the waveform vector.

The elements of a WaveformVectorT type are described in Table A.85.

Table A.85 – Elements of WaveformVectorT

Element	Description
WaveformVectorElementList	WaveformVectorElementList contains static information about each individual element of a waveform vector.

A.89 WaveformVectorElementListT

It represents the static information of a waveform vector, except the nodepath for the array of scalar values. This is the additional information such as Handling, Range, Type.

The XML schema for a WaveformVectorElementListT type is:

```
<xs:complexType name="WaveformVectorElementListT">
   <xs:sequence>
      <xs:element name="WaveformVectorElement" type="clnt:WaveformVectorElementT"
         minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
</xs:complexType>
```

The elements of a WaveformVectorElementListT type are described in Table A.86.

Table A.86 – Elements of WaveformVectorElementListT

Element	Description
WaveformVectorElement	The static information of an individual waveform vector element.

A.90 WaveformVectorElementT

It represents the whole information of a single element in a waveform vector. It contains additional information about one value such as Handling, Range, Type. It belongs to the corresponding element in the result array of DataArrayNodePath.

The XML schema for a WaveformVectorElementT type is:

```
<xs:complexType name="WaveformVectorElementT">
   <xs:sequence>
      <xs:element name="DataType" type="clnt:NumericDataT" default="SignedInteger"
         minOccurs="0"/>
      <xs:element name="RangeList" type="clnt:RangeListT" minOccurs="0"/>
      <xs:element name="Handling" type="clnt:HandlingT" default="rw" minOccurs="0"
         maxOccurs="1"/>
      <xs:element name="PreEditActionsList" type="clnt:ActionListT" minOccurs="0"
         maxOccurs="1"/>
      <xs:element name="PostEditActionsList" type="clnt:ActionListT" minOccurs="0"
         maxOccurs="1"/>
      <xs:element name="ScalingFactor" type="xs:double" minOccurs="0"/>
      <xs:element name="DisplayFormat" type="clnt:FormatSpecifierT" minOccurs="0"/>
      <xs:element name="EditFormat" type="clnt:FormatSpecifierT" minOccurs="0"/>
   </xs:sequence>
</xs:complexType>
```

The elements of a WaveformVectorElementT type are described in Table A.87.

Table A.87 – Elements of WaveformVectorElementT

Element	Description
DataType	This required element specifies the data type of the value in the corresponding element of the result array of DataArrayNodePath.
RangeList	This optional element specifies the acceptable range of the value in the corresponding element of the result array of DataArrayNodePath.
Handling	This optional element specifies whether or not the current data value may be modified. The default is ReadWrite.
PreEditActionsList	This optional element specifies the PreEditActions of the value in the corresponding element of the result array of DataArrayNodePath.
PostEditActionsList	This optional element specifies the PostEditActions of the value in the corresponding element of the result array of DataArrayNodePath.
ScalingFactor	This optional element specifies the scaling factor of the value in the corresponding element of the result array of DataArrayNodePath.
DisplayFormat	This optional element specifies the display format of the value in the corresponding element of the result array of DataArrayNodePath.
EditFormat	This optional element specifies the edit format of the value in the corresponding element of the result array of DataArrayNodePath.

Annex B
(informative)

Action example

The following is an example of an EDD method being executed by an FDI Server and the resulting interaction with an FDI Client.

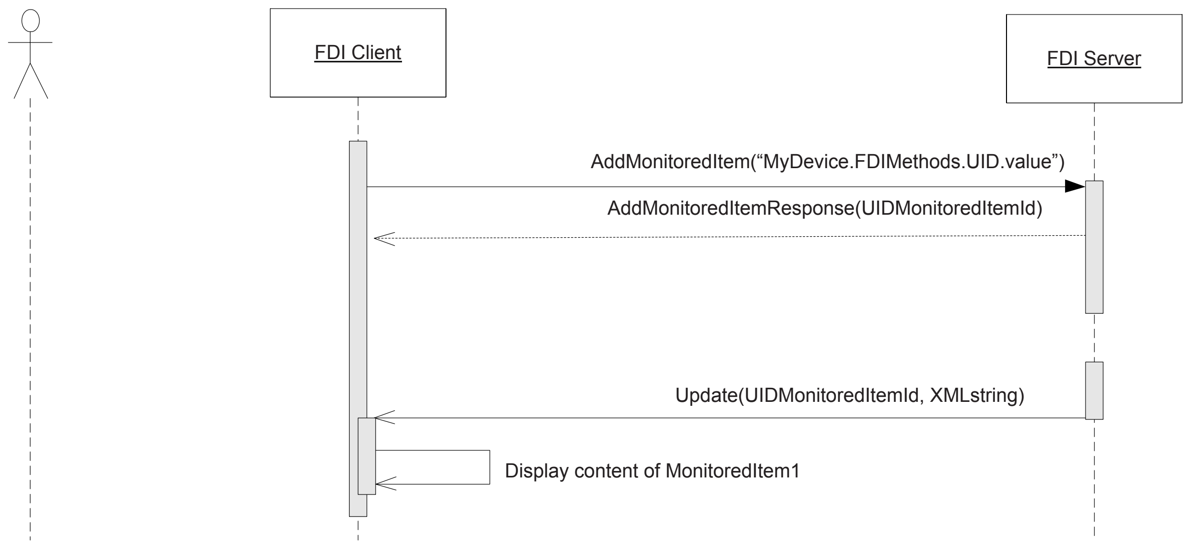
The EDDL used by this example is shown below, followed by a sequence diagram and the description of the sequence.

```
VARIABLE device_var1
{
    LABEL "device var1";
    HELP  "";
    CLASS DEVICE;
    HANDLING READ & WRITE;
    CONSTANT_UNIT "constUnit";
    TYPE INTEGER;
    PRE_EDIT_ACTIONS
    {
        PreEditAction1
    }
    POST_EDIT_ACTIONS
    {
        PostEditAction1
    }
}

VARIABLE process_value
{
    LABEL "Level";
    HELP  "";
    CLASS DYNAMIC;
    HANDLING READ;
    CONSTANT_UNIT "m";
    TYPE FLOAT;
}

VARIABLE newI
{
    LABEL "new i";
    HELP  "new value of i";
    CLASS DEVICE;
    HANDLING READ & WRITE;
    TYPE INTEGER;
}

VARIABLE newJ
{
    LABEL "new j";
    HELP  "new value of j";
    CLASS DEVICE;
    HANDLING READ & WRITE;
    TYPE INTEGER;
}

MENU MethodMenu
{
    LABEL "MethodMenu";
    HELP  "This menu is used in a method";
    STYLE DIALOG;
    ITEMS
    {
        newI,
        newJ
    }
```

```
}


METHOD  UIReqRespCategories
{
    LABEL "Request Response Categories";
    HELP "This method demonstrates different categories of messages that are passed
          between server and client during a method execution";
    DEFINITION
    {
        int i, j, k, selection;
        add_abort_method(AbortMethod);
        //Acknowledgement
        ACKNOWLEDGE("Please hit OK to acknowledge the start of this method");
                                                                        // A010
        i = 5;
        PUT_MESSAGE("i = %{i}"); //Info // A020
        j = 10;
        PUT_MESSAGE("j = %{j}"); //Info // A030
        k = i+j;
        DELAY(5, "k = %{k}");  //Delay // A040

        GET_DEV_VAR_VALUE("Enter new value for the
                    %[L]%[U]{device_var1}%[U]",device_var1); //Input // A050

        selection = SELECT_FROM_LIST("Is the value entered, correct? ", "YES;NO");
                                                          //selection // A060
        if (selection == 1)
        {
            device_var1 = 0;
            abort(); //abort --- this will trigger the AbortMethod as well; // A070
        }
        k = k + device_var1;

        if (k == i + j)
        {
            display_comm_status(2);//Error - 2 == "Buffer Overflow" -- HART;
                                                                        // A080
        }

        display("Current level is %{process_value}%[U]{process_value}!"); // A090
        MenuDisplay(MethodMenu,"APPLY;DISCARD",selection); //UIDMessage // A100
        if(selection == 0)
        {
            i = newI;
            j = newJ;
            ACKNOWLEDGE("new value of k = %{k}"); // A110
        }
       ACKNOWLEDGE("This concludes the method !!"); // A120
    }
}

METHOD AbortMethod
{
    LABEL "AbortMethod";
    HELP  "This is a simple Abort Method";
    DEFINITION
    {
      ACKNOWLEDGE("Method was aborted due to a call to abort()"); // B010
    }
}

METHOD PreEditAction1
{
    LABEL "Action1";
    HELP  "This is a simple Pre Edit Warning";
    DEFINITION
    {
      ACKNOWLEDGE("Do you really want to edit this variable"); // C010
    }
}

METHOD PostEditAction1
{
    LABEL "Action2";
    HELP  "This is a simple Post Edit Message";
    DEFINITION
    {
      ACKNOWLEDGE("You actually edited the variable now!!!"); // D010
    }
}

MENU FDIActions
{
    LABEL "FDI Actions";
    HELP "This menu contains methods for verifying FDI UID contents";
    ITEMS
    {
        MethodMenu,
        UIReqRespCategories
    }
}
```

The example assumes as a precondition that the FDI Client has already established a Session with the FDI Server, the user has navigated to the Device (MyDevice), and the user has initiated the opening of the function group (FDIActions).

The sequence begins with the FDI Client subscribing to the value of the UID in order to retrieve the UID content from the FDI Server. The FDI Client adds a monitored item and waits for the user to provide the initial value. The sequence is illustrated in Figure B.1 along with the XML string that will be returned from the FDI Server.


FDI Client
FDI Server
AddMonitoredItem("MyDevice.FDIMethods.UID.value")
AddMonitoredItemResponse(UIDMonitoredItemId)
Update(UIDMonitoredItemId, XMLstring)
Display content of MonitoredItem1

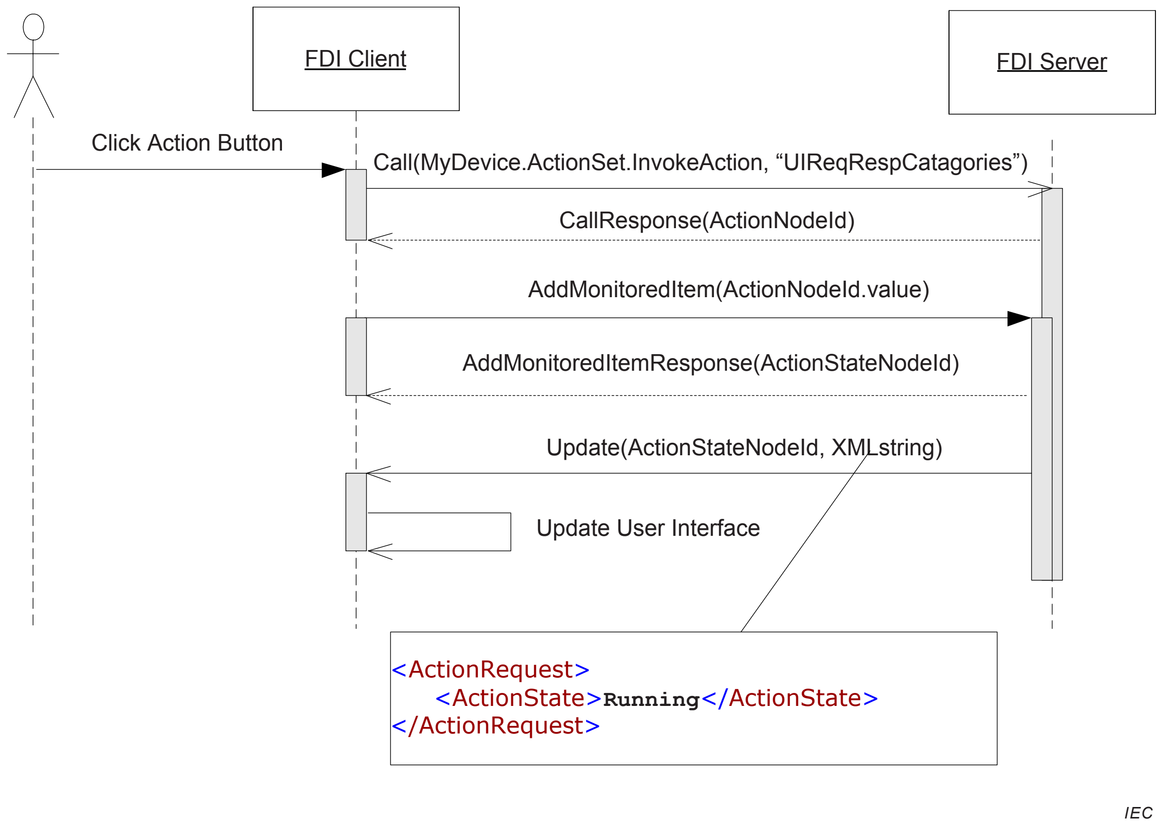
```
<Window>
  <Label>FDI Actions</Label>
  <Help>This menu contains methods for verifying FDI UID contents</Help>
  <Items>
    <Dialog NodePath="/MethodMenu">
      <Label>MethodMenu</Label>
      <Help>This menu is used in a method</Help>
    </Dialog>
    <Action>
      <Name>UIReqRespCategories</Name>
      <Label>Request Response Categories</Label>
      <Help>This method demonstrates different categories…</Help>
    </Action>
  </Items>
</Window>
```

IEC

Figure B.1 – Action example (step 1)

The FDI Client interprets the XML string and presents the menu to the user as four action buttons with each button containing the text defined in the <Label> element. The FDI Client then waits for the user to select one of the actions.

In the next step of the example the sequence begins with the user clicking the action button (UIReqRespCategories) presented in the menu. The FDI Client reacts by initiating an OPC UA Call service request (see IEC 62541-4), specifying the OPC UA method (MyDevice.ActionSet.InvokeAction) and the EDDL method (UIReqRespCategories). The FDI Server responds by locating the EDDL method and initiating execution of it. The FDI Server responds with a unique node Id (Action NodeId1) that represents the running method. The FDI Client subscribes to the value of the method using the node Id (ActionNodeId.value). The FDI Server provides the current value of the method as any XML string. The initial value of this XML string indicates that the method has started. The sequence of this step is illustrated in Figure B.2.


FDI Client
FDI Server
Click Action Button
Call(MyDevice.ActionSet.InvokeAction, “UIReqRespCatagories”)
CallResponse(ActionNodeId)
AddMonitoredItem(ActionNodeId.value)
AddMonitoredItemResponse(ActionStateNodeId)
Update(ActionStateNodeId, XMLstring)
Update User Interface
IEC

```
<ActionRequest>
    <ActionState>Running</ActionState>
</ActionRequest>
```

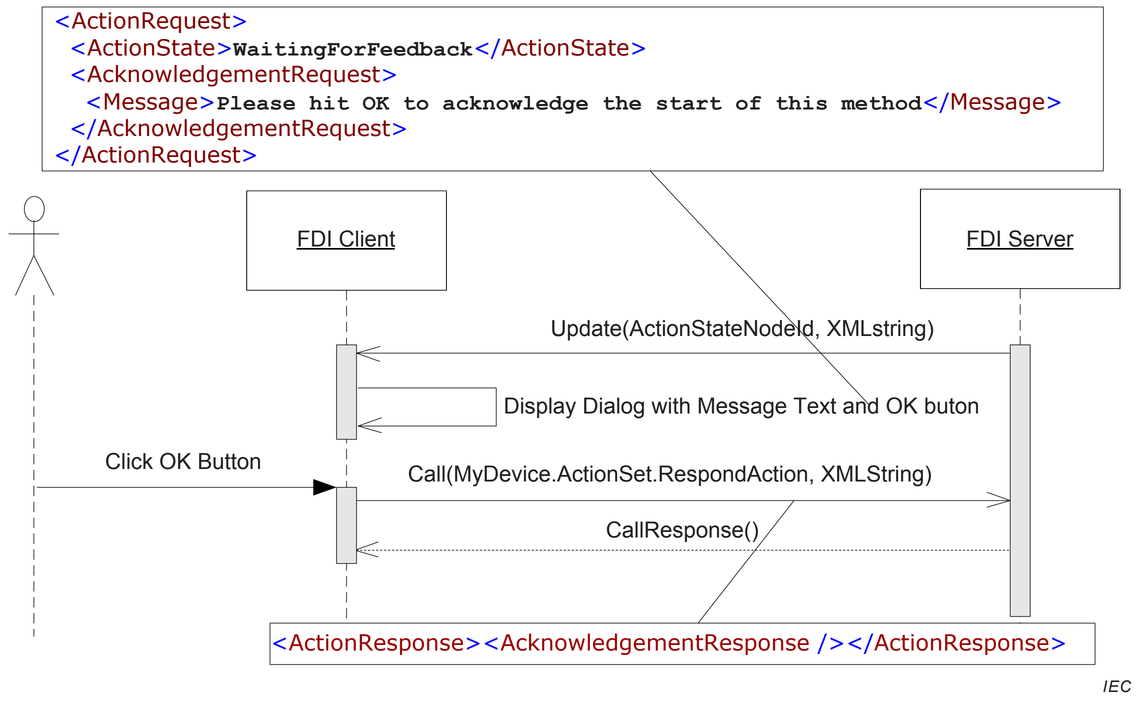
Figure B.2 – Action example (step 2)

In the next step of the example the method has run up to the “ACKNOWLEDGE” statement. The FDI Server processes the statement by setting the ActionNodeID value to indicate that a user acknowledgement is needed before the method can continue. The change to the value results in the FDI Server issuing a subscription update. Upon receiving the value update the FDI Client detects that an acknowledge request is pending. The FDI Client reacts by opening a dialog to present the message received in the value update also with an OK button.

The user reads the message and presses the OK button resulting in the FDI Client issuing a RespondAction method call to the FDI Server.

The FDI Server, receiving the RespondAction call, continues the execution of the method.

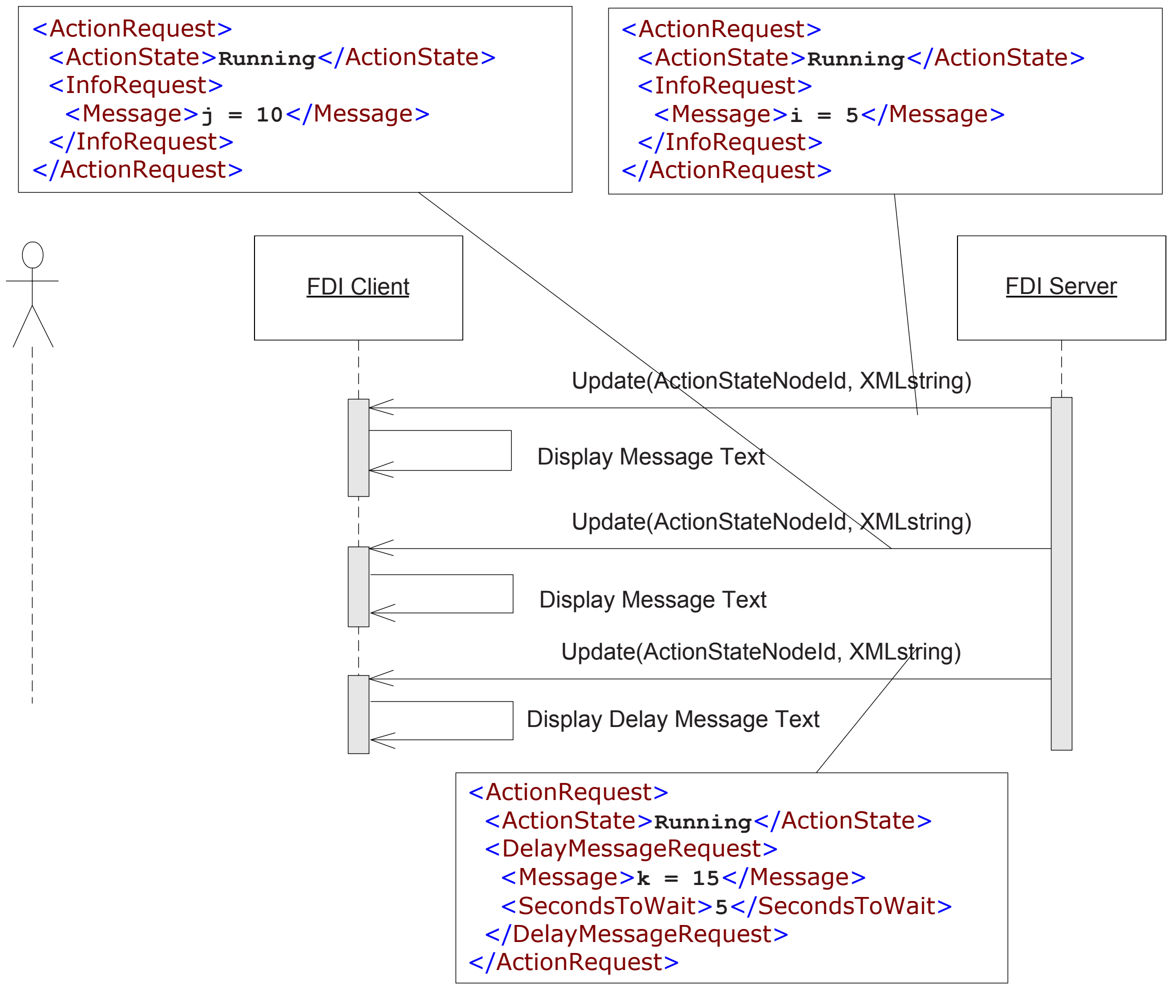
The sequence of this step is illustrated in Figure B.3.



<ActionRequest>
<ActionState>WaitingForFeedback</ActionState>
<AcknowledgementRequest>
<Message>Please hit OK to acknowledge the start of this method</Message>
</AcknowledgementRequest>
</ActionRequest>
FDI Client
FDI Server
Update(ActionStateNodeId, XMLstring)
Display Dialog with Message Text and OK buton
Click OK Button
Call(MyDevice.ActionSet.RespondAction, XMLString)
CallResponse()
<ActionResponse><AcknowledgementResponse /></ActionResponse>
IEC

Figure B.3 – Action example (step 3)

In the next step of the example the method has just executed the “ACKNOWLEDGE” statement and is about to execute the two “PUT_MESSAGE” statements and the “DELAY” statement. The FDI Server processes these statements by setting the ActionNodeID value to indicate the notifications. Each statement will generate a value change and be delivered to the FDI Client. The FDI Client reacts by showing the message text in an appropriate user interface component. The sequence of this step is illustrated in Figure B.4.


FDI Client
FDI Server
Update(ActionStateNodeId, XMLstring)
Display Message Text
Update(ActionStateNodeId, XMLstring)
Display Message Text
Update(ActionStateNodeId, XMLstring)
Display Delay Message Text
IEC

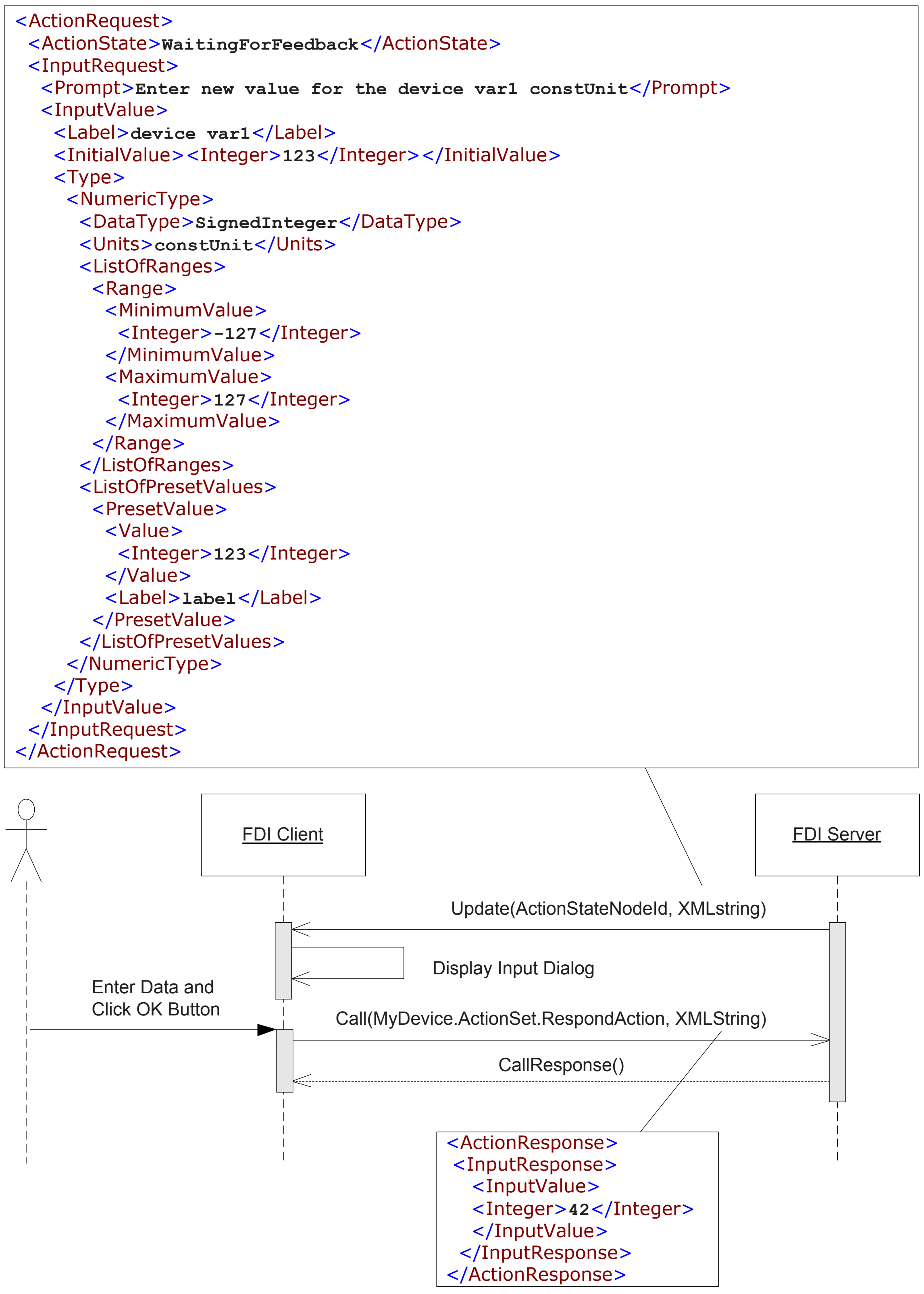
```
<ActionRequest>
  <ActionState>Running</ActionState>
  <InfoRequest>
    <Message>j = 10</Message>
  </InfoRequest>
</ActionRequest>
```

```
<ActionRequest>
  <ActionState>Running</ActionState>
  <InfoRequest>
    <Message>i = 5</Message>
  </InfoRequest>
</ActionRequest>
```

```
<ActionRequest>
  <ActionState>Running</ActionState>
  <DelayMessageRequest>
    <Message>k = 15</Message>
    <SecondsToWait>5</SecondsToWait>
  </DelayMessageRequest>
</ActionRequest>
```

Figure B.4 – Action example (step 4)

In the next step of the example the method requests the user to input a value for the (device_var1) variable. The FDI Server processes this statement by setting the ActionNodeID value to indicate the data input request. The FDI Client reacts to the value update by opening a dialog, showing the message text and preparing to accept user input. The sequence of this step is illustrated in Figure B.5.

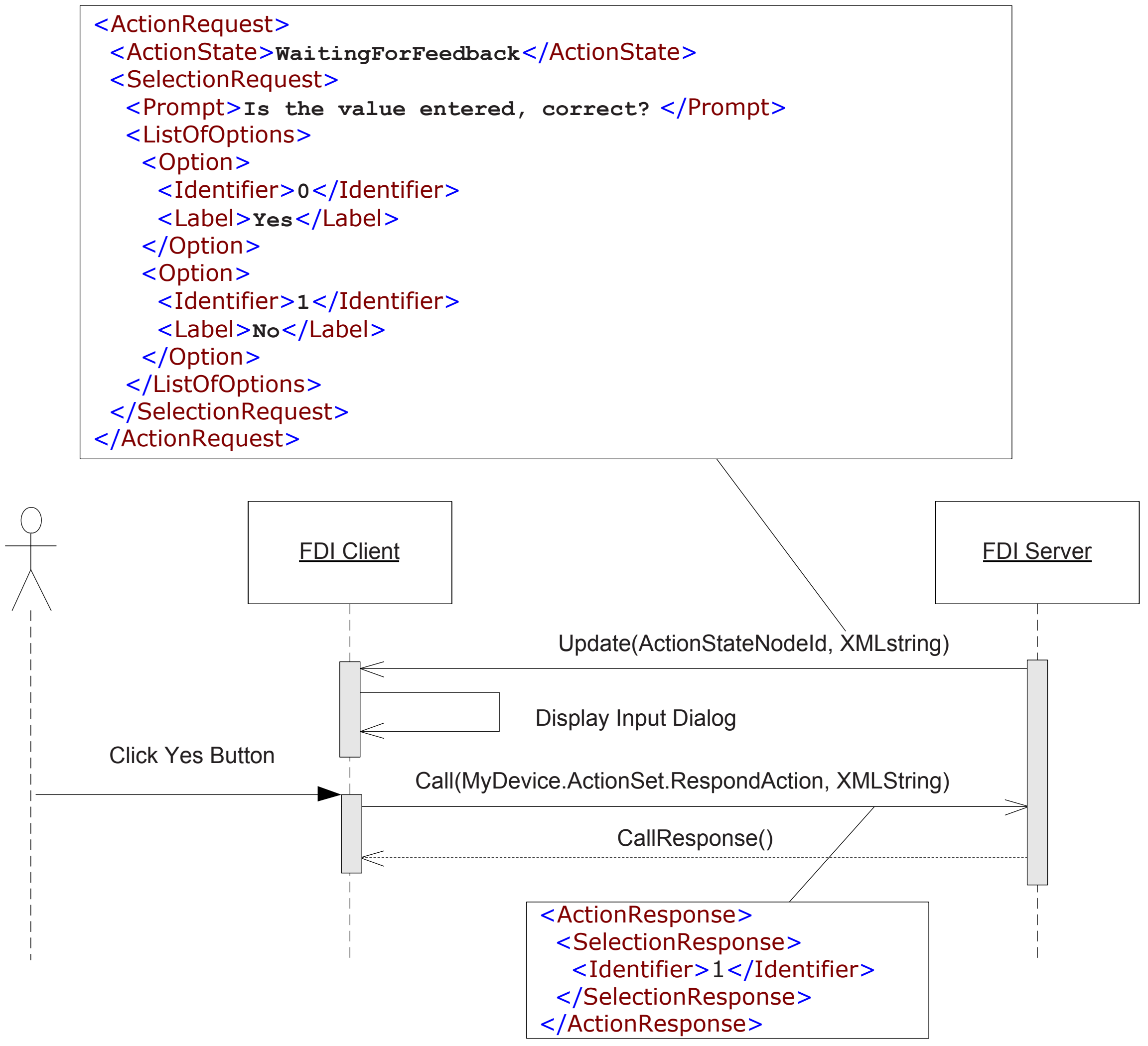


```
<ActionRequest>
 <ActionState>WaitingForFeedback</ActionState>
 <InputRequest>
  <Prompt>Enter new value for the device var1 constUnit</Prompt>
  <InputValue>
   <Label>device var1</Label>
   <InitialValue><Integer>123</Integer></InitialValue>
   <Type>
    <NumericType>
     <DataType>SignedInteger</DataType>
     <Units>constUnit</Units>
     <ListOfRanges>
      <Range>
       <MinimumValue>
        <Integer>-127</Integer>
       </MinimumValue>
       <MaximumValue>
        <Integer>127</Integer>
       </MaximumValue>
      </Range>
     </ListOfRanges>
     <ListOfPresetValues>
      <PresetValue>
       <Value>
        <Integer>123</Integer>
       </Value>
       <Label>label</Label>
      </PresetValue>
     </ListOfPresetValues>
    </NumericType>
   </Type>
  </InputValue>
 </InputRequest>
</ActionRequest>
```

```
<ActionResponse>
 <InputResponse>
   <InputValue>
   <Integer>42</Integer>
   </InputValue>
  </InputResponse>
</ActionResponse>
```

Figure B.5 – Action example (step 5)

In the next step of the example the method requests the user to verify the value that was inputted for the (device_var1) variable. The FDI Server processes this statement by setting the ActionNodeID value to indicate a response is needed. The FDI Client reacts to the value update by opening a dialog, showing the message text and preparing to accept a response from the user. The sequence of this step is illustrated in Figure B.6.



```xml
<ActionRequest>
  <ActionState>WaitingForFeedback</ActionState>
  <SelectionRequest>
    <Prompt>Is the value entered, correct? </Prompt>
    <ListOfOptions>
      <Option>
        <Identifier>0</Identifier>
        <Label>Yes</Label>
      </Option>
      <Option>
        <Identifier>1</Identifier>
        <Label>No</Label>
      </Option>
    </ListOfOptions>
  </SelectionRequest>
</ActionRequest>
```

```xml
<ActionResponse>
  <SelectionResponse>
    <Identifier>1</Identifier>
  </SelectionResponse>
</ActionResponse>
```

FDI Client
FDI Server
Update(ActionStateNodeId, XMLstring)
Display Input Dialog
Click Yes Button
Call(MyDevice.ActionSet.RespondAction, XMLString)
CallResponse()
IEC

Figure B.6 – Action example (step 6)

Annex C
(informative)

Typical FDI Client use cases

C.1 General

Annex C describes how typical FDI Client use cases could be implemented.

C.2 Bulk operations

In Clause C.2 the term bulk operation is understood as applying the same operation iteratively to a group of Devices. If the group of Devices consists of Devices of different Device Types, a precondition is the clarification of what the “same” operations are with respect to the different Device Types.

Bulk operations can be performed by an FDI Client or by a UIP. In case of a UIP the group of devices shall be a subset of the Device and its sub-devices because a UIP has only access to these.

NOTE The following description only refers to an FDI Client. The algorithm is the same for a UIP besides the above mentioned restriction.

First of all an FDI Client has to give the user the ability to define a group of Devices. This is implementation specific and outside the scope of this document. The FDI Client can then use the OPC UA services to apply the same operation to these Devices. For that, OPC UA supports reading and writing of multiple parameters in one service call as well as invoking several methods in one service call. See IEC 62541-4 for details.

C.3 Progress bar support

The following is an example of a UIP using the Progress Bar functionality provided by the FDI Client (see Figure C.1). In this example the user initiates some operations by interacting with the UIP’s user interface. The UIP initiates the operation and uses the progress bar functionality to keep the user informed of the progress (see 5.2.2.10 and 5.2.2.11).

In this example the FDI Client opens a small dialog window to present the progress information to the user. Once the UIP informs the FDI Client that the operation is complete, via the EndShowProgressBar service (see 5.2.2.12), the FDI Client prompts the user before closing the dialog to ensure the user has seen the 100 % completion of the operations.

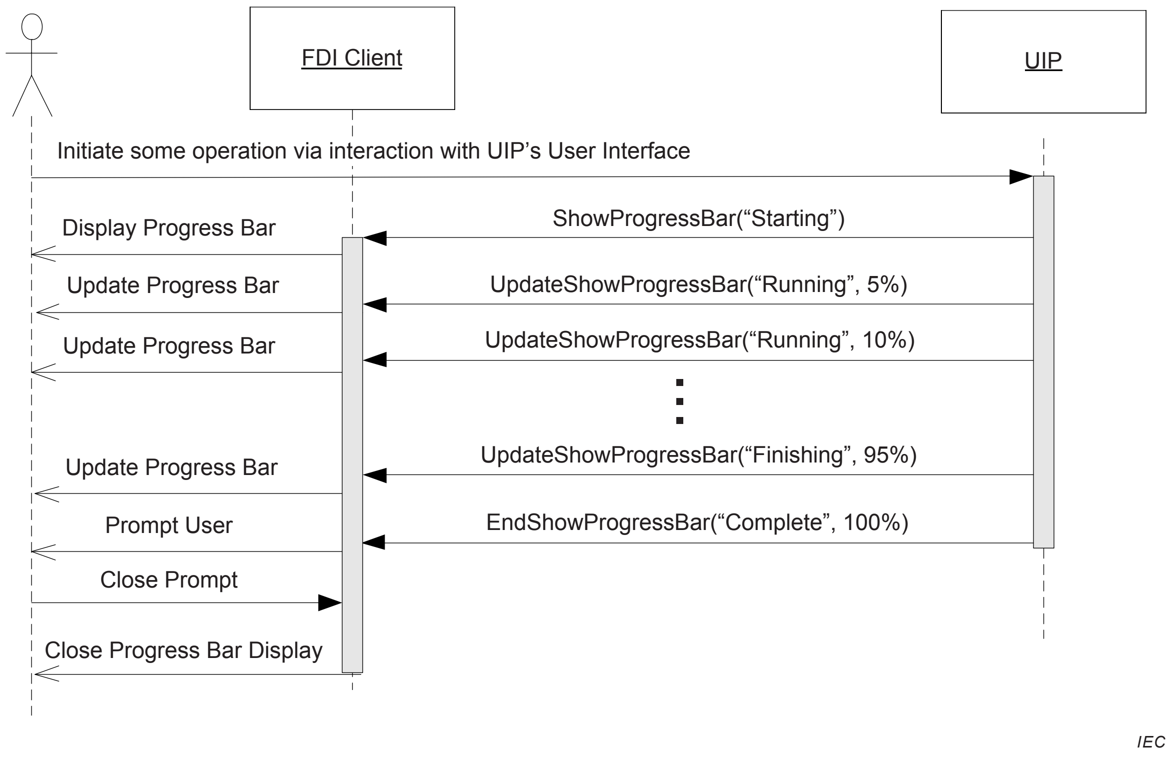

FDI Client
UIP
Initiate some operation via interaction with UIP's User Interface
ShowProgressBar("Starting")
Display Progress Bar
UpdateShowProgressBar("Running", 5%)
Update Progress Bar
UpdateShowProgressBar("Running", 10%)
Update Progress Bar
UpdateShowProgressBar("Finishing", 95%)
Update Progress Bar
EndShowProgressBar("Complete", 100%)
Prompt User
Close Prompt
Close Progress Bar Display
IEC

Figure C.1 – Progress bar support

Bibliography

ISO/IEC 10646-1*, Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane*

ISO/IEC 2375*, Information technology – Procedure for registration of escape sequences and coded character sets*

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

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

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

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

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

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

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

___

This page deliberately left blank

British Standards Institution (BSI)

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

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

About us

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

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

Information on standards

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

Buying standards

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

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

Subscriptions

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

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

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

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

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

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

Revisions

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

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

Copyright

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

Useful Contacts:

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

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

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

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

BSI Group Headquarters

389 Chiswick High Road London W4 4AL UK

bsi.

...making excellence a habit.™