IEC 62453-2

Edition 3.0 2022-08

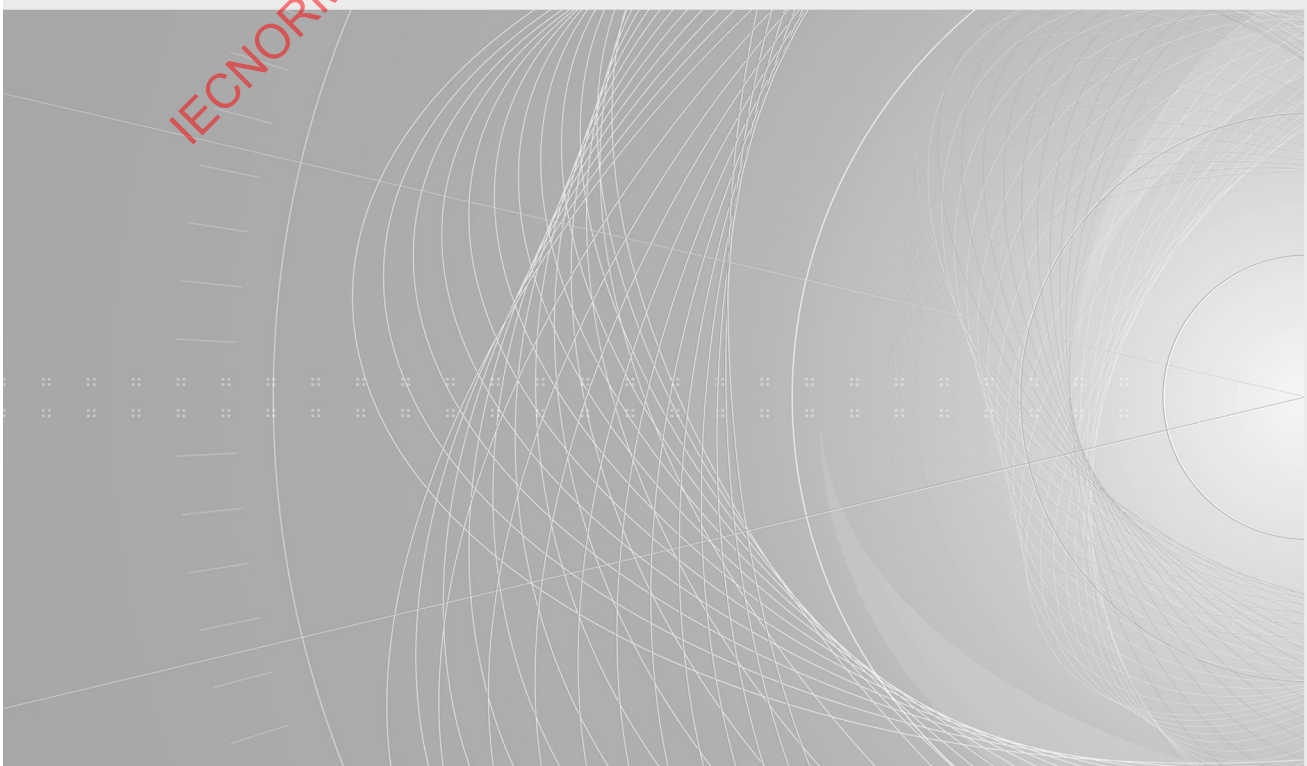
INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Field device tool (FDT) interface specification – Part 2: Concepts and detailed description

Spécification des interfaces des outils des dispositifs de terrain (FDT) – Partie 2: Concepts et description détaillée



IEC 62453-2:2022-08(en-fr)

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

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

IEC 62453-2

Edition 3.0 2022-08

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Field device tool (FDT) interface specification – Part 2: Concepts and detailed description

Spécification des interfaces des outils des dispositifs de terrain (FDT) – Partie 2: Concepts et description détaillée

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

ICS 25.040.40; 35.100.05; 35.110

ISBN 978-2-8322-4532-3

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

FIELD DEVICE TOOL (FDT) INTERFACE SPECIFICATION –

Part 2: Concepts and detailed description

FOREWORD

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

IEC 62453-2 has been prepared by subcommittee 65E: Devices and integration in enterprise systems, of IEC technical committee 65: Industrial-process measurement, control and automation. It is an International Standard.

This third edition cancels and replaces the second edition published in 2016. This edition constitutes a technical revision.

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

a) clarification for Static Function,
b) clarification regarding system GUI label,
c) clarification regarding loss of connection.

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

Draft	Report on voting
65E/906/FDIS	65E/933/RVD

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

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

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

A list of all parts of the IEC 62453 series, under the general title *Field device tool (FDT) interface specification*, can be found on the IEC website.

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

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

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

INTRODUCTION

This part of IEC 62453 is an interface specification for developers of FDT[1] (Field Device Tool) components for function control and data access within a client/server architecture. The specification is a result of an analysis and design process to develop standard interfaces to facilitate the development of servers and clients by multiple vendors that need to interoperate seamlessly.

With the integration of fieldbuses into control systems, there are a few other tasks which need to be performed. In addition to fieldbus- and device-specific tools, there is a need to integrate these tools into higher-level system-wide planning or engineering tools. In particular, for use in extensive and heterogeneous control systems, typically in the area of the process industry, the unambiguous definition of engineering interfaces that are easy to use for all those involved is of great importance.

A device-specific software component created according to this document is called Device Type Manager (DTM). It integrates all device-specific data, functions and business rules into the system via the FDT services defined herein.

The FDT/DTM approach is open for all kind of fieldbuses and enables integration variety of devices into heterogeneous systems.

Figure 1 shows how this document is aligned in the structure of the IEC 62453 series.

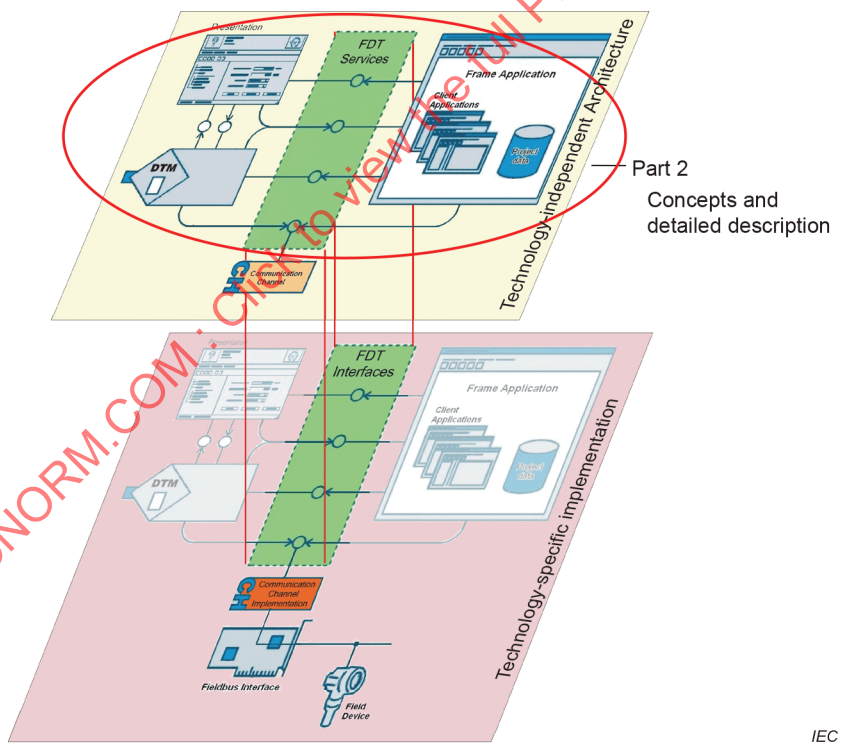


Figure 1 – Part 2 of the IEC 62453 series

[1] FDT® is a trademark of products supplied by FDT Group AISBL. This information is given for convenience of users of this document and does not constitute an endorsement by IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

FIELD DEVICE TOOL (FDT) INTERFACE SPECIFICATION –

Part 2: Concepts and detailed description

1 Scope

This part of IEC 62453 explains the common principles of the field device tool concept. These principles can be used in various industrial applications such as engineering systems, configuration programs and monitoring and diagnostic applications.

This document specifies the general objects, general object behavior and general object interactions that provide the base of FDT.

2 Normative references

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

IEC 61131 (all parts), *Programmable controllers*

IEC TR 62390:2005, *Common automation device – Profile guideline*

IEC 62453-1:2016, *Field device tool (FDT) interface specification – Part 1: Overview and guidance*

IEC 62453-3xy (all parts), *Field device tool (FDT) interface specification – Part 3xy: Communication profile integration*

3 Terms, definitions, symbols, abbreviated terms and conventions

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 62453-1 as well as the following apply.

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

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

3.1.1
Child DTM
DTM instance in an FDT Project, which is classified by its relation to a Parent DTM

Note 1 to entry: Any DTM which uses FDT communication may be classified as Child DTM (i.e. Device DTM, Gateway DTM, Module DTM and BTM).

3.1.2
FDT version
implementation version defined by the related technology-specific organization

Note 1 to entry: The FDT version is specified in IEC TR 62453-41 or in IEC TR 62453-42.

3.1.3
monolithic DTM
one single DTM that represents the complete device with all its modules

Note 1 to entry: There are also other concepts representing modules of a device in this specification, for example Module DTM and BTM.

3.1.4
Parent DTM
DTM instance in an FDT Project, which is classified by its relation to a Child DTM

Note 1 to entry: Any DTM which provides FDT communication may be classified as Parent DTM (e.g. Communication DTM, Gateway DTM, Composite Device DTM).

3.2 Symbols and abbreviated terms

For the purposes of this document, the symbols and abbreviated terms given in IEC 62453-1 as well as the following apply.

DD	Device description
OODMS	Object Oriented Database Management System
RDBMS	Relational Database Management System

3.3 Conventions

3.3.1 Use of UML

The conventions for UML notation used in this document are defined in IEC 62453-1.

3.3.2 State availability statement

The state availability statement uses [_] and [X] for stating the availability of a service in a specific state. The expressions have the following meaning:

[_] '<name of state>' service is not available in this state;
[X] '<name of state>' service is available in this state.

3.3.3 Data type names and references to data types

The conventions for naming and referencing data types are explained in Clause A.1.

4 Fundamentals

4.1 General

Clause 4 introduces the FDT model and covers topics which are specific to the requirements of field device integration.

4.2 Abstract FDT model

4.2.1 FDT model overview

The UML class diagram in Figure 2 provides an overview of the objects defined in FDT and their relationships to each other.

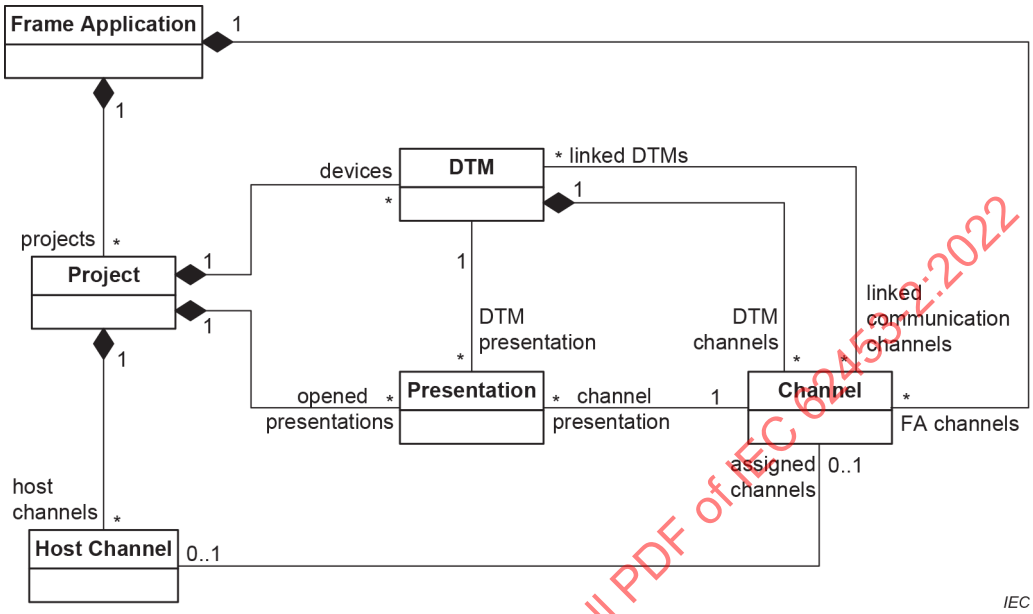


Figure 2 – Abstract FDT model

Table 1 contains brief descriptions of all objects. A more detailed description of objects and related services is provided in 4.2.2 to 4.2.4.

The FDT objects may be executed all on one platform or in a distributed system.

For data exchange and for the interaction between objects, the data types as defined in Annex A are used. The concrete implementation of these data types is defined in the object model integration profile documents of the IEC 62453 series (e.g. IEC TR 62453-41, IEC TR 62453-42) describing the mapping to specific implementation technologies.

Table 1 – Description of FDT objects

Object	Description
Frame Application	The Frame Application is a logical object to represent an environment like an engineering system or a standalone tool (see 4.2.2). It controls the lifetime of DTM-instances within the Project. A Frame Application may handle several Projects.
Project	The Project belongs to the Frame Application. The Project is a logical object describing the management of device-instances. It controls the lifetime and dataset of device-instances within a Frame Application. FDT does not define any services for the Project, since it is a pure Frame Application internal object.
Host Channel	A Host Channel belongs to the Frame Application. The Host Channel is a logical object representing a part of a function of a host system, such as DCS or PLC. It is required when additional information for the processing of device I/O data is needed, e.g. if DTM or BTM data refers to more than one I/O value. For example one Octet is often used to group the state values of eight digital I/O points. In these applications, the Host Channel object provides an association between the contents of the channel data and individual process I/O points. To make the cyclic I/O data available within a Frame Application, there shall be an association between the I/O function blocks and the process values of a device. The inputs and outputs of I/O function blocks are represented by Host Channels, the process value is represented by a Channel. The association between Host Channel and Channel is called channel assignment. FDT does not define any services for the Host Channel, since it is a pure Frame Application internal object.
DTM	A DTM is a software component containing the device-specific application software. It encapsulates all device-specific data, functions and business rules. A DTM is generally not stand-alone executable. It needs a Frame Application (see 4.2.2) that acts as the runtime environment. A DTM is typically developed by the device manufacturer and shipped together with the devices. It depends on the software design of the DTM, whether it can be used for one specific device type or a group of different device types or a device type family. Each device installed on a plant is represented by a DTM object. Therefore one or several DTM objects are needed to handle the different devices. The DTMs are installed in the system, so that the system can be dynamically extended by installing new DTMs for new devices. A DTM may represent different types of devices, e.g. field devices, communication devices or gateway devices (see 4.2.3).
Presentation	Presentation objects (see 4.2.3.3) represent a graphical user interface (GUI). The Presentation object belongs to the DTM, to the BTM or to the Channel.
Channel	Channel object could behave in three ways: As a 'Communication Channel', a 'Process Channel' and a combination of both (see 4.2.4).

All the associations shown in Figure 2 are explained in Table 2.

Table 2 – Description of associations between FDT objects

Association	Description
assigned channels	To make the cyclic I/O data available within a Frame Application, there has to be an association between the I/O function blocks and the process values of a device. The I/O function blocks are represented by a Host Channel, the process value by a channel. The Frame Application (Project) is responsible for handling this association. Use cases: – channel assignment Services: Informational services – DTM service: GetChannels – channel service: ReadChannelInformation Management services – channel service: WriteChannelInformation Depending services – proprietary DTM services for channel management
channel presentation	The instances of Presentation objects are associated with the instance of their DTM business object by this relationship. Use cases: – all channel use cases with DTM-specific GUI Services: Informational services (see also 7.3) – channel service: GetChannelFuntions – channel service: GetGuiInformation Depending services – frame service: OpenPresentationRequest – frame service: ClosePresentationRequest
devices	A Project holds the list of DTM instances by the association devices. Releasing the Project results in a release of all associated DTM instances. A DTM instance cannot be part of more than one Project. Use cases: – system generation – system planning Services: Informational services – frame service: GetChildNodes Management services – services related to sub-topology management, see 7.6.2 – ValidateAddChild – ChildAdded – ValidateRemoveChild – ChildRemoved – Frame Application services related to topology management
DTM channels	A DTM can have a list of channels, which are associated with DTM channels. A channel is part of a DTM. Services: Informational services – frame service: GetChannels Management services – There are no services to manipulate this association. The lifetime of channel instances is controlled by a DTM

Association	Description
DTM presentation	The Presentation object instances are associated with their DTM instance by this relationship. Use cases: – All DTM use cases with DTM-specific GUI Services: – open presentation – close presentation
FA channels	The Frame Application (project) may have a list of Communication Channels. There are no services to manipulate this association. See 4.2.2.
Host Channels	The Frame Application maps the Process Channels of DTMs to internal I/O management of the projects.
linked communication channels	The topology of DTMs is shown with this association. Within the topology tree, a DTM object is the child node of a channel. The association shows the connection from a device to a channel. The Frame Application (project) is responsible for handling this association. Services: – linked channels can be explored by GetParentNodes – SetLinkedCommunicationChannel – ReleaseLinkedCommunicationChannel
linked DTMs	The topology of field devices is shown with this association. Within the topology tree, a Channel object is the parent node for a device. The association shows the connection from a channel to a device. The Frame Application (Project) is responsible for handling this association. Services: – linked DTMs can be explored by GetChildNodes – SetLinkedCommunicationChannel – ReleaseLinkedCommunicationChannel
opened presentations	Open Presentation objects are linked by this association to Projects. The Frame Application (Project) is responsible for handling this association.
projects	A Frame Application can create and manage multiple Projects, which are connected by the "projects" association. The Frame Application (Project) is responsible for handling this association.

4.2.2 Frame Application (FA)

4.2.2.1 General

Frame Applications are able to handle any type of device by integrating corresponding DTMs without the need for device-specific or fieldbus-specific knowledge. Depending on the intended use, Frame Applications may provide different appearance and features (e.g. standalone configuration or engineering system of a DCS). Typically, the Frame Application comprises client applications focusing on specific aspects such as configuration, observation, I/O planning, and using the service provided by the DTMs.

The Frame Application is the runtime environment for the DTMs. It provides the services described in 7.7 which may be used by the hosted DTMs.

4.2.2.2 System communication

If the Frame Application has built-in communication capabilities, then it is able to provide Communication Channels (see 4.2.4.3) which provide the services to access the fieldbus. In this case it has full control over communication and is able perform low-level actions such as monitoring or filtering. Frame Applications without build-in communication capabilities are using Communication DTMs (see 4.2.3.2.2). Figure 3 shows the relation between the Frame Application and the Communication Channel object.

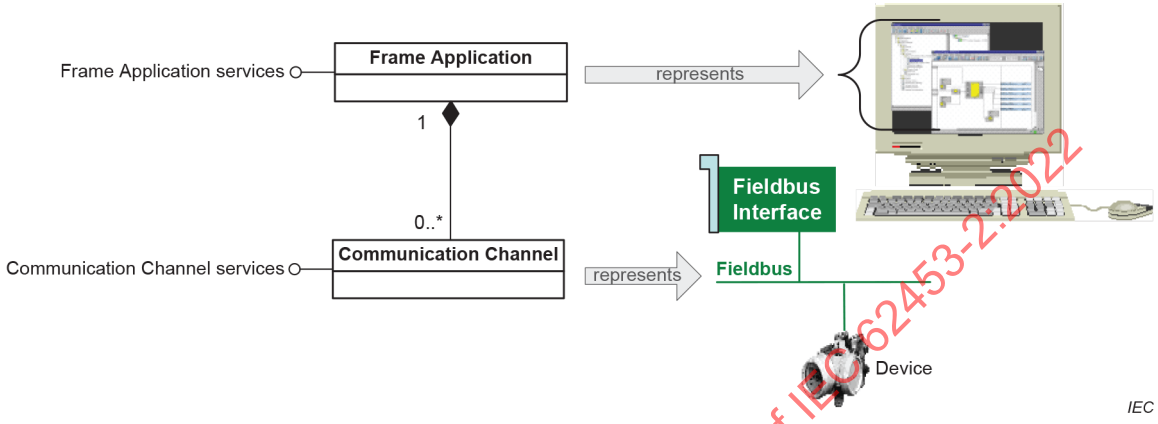


Figure 3 – Frame Application with integrated Communication Channel

The Frame Application's Channel objects represent the gateway from the FDT-specific to the Frame Application-specific communication. On the Frame Application's side, the Communication Channel may be a PC I/O board or an engineering topology with processing units and proprietary bus systems.

4.2.3 Device Type Manager (DTM)

4.2.3.1 DTM Business Logic overview

The DTM Business Logic (DTM BL) encapsulates device-specific functions and protocol-specific definitions. Thus a Frame Application is able to handle any type of device by integrating corresponding DTM Business Logic without the need for device- or fieldbus-specific knowledge.

The DTM BL provides the services described in 7.2 (see Figure 4). From a Frame Application's point of view, all management and task-related interactions with the device functionality are available via these services.

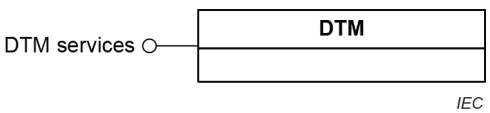


Figure 4 – Device Type Manager (DTM)

A DTM may represent the device structure with data type net:DtmDeviceInstanceTopology (see Clause A.12). This information may be retrieved by service GetActiveTypeInfo (see 7.2.3.4). The structure of the device may be described by a list of net:Module. Each module may have a number of properties, including configuration data, Process Channels and Communication Channels.

A DTM exposes a list of supported device types with the data type fdt:DtmDeviceTypes (see 4.8.1).

4.2.3.2 Categories of DTM

4.2.3.2.1 General

DTMs represent measurement and control devices, modules, gateways or blocks as well as communication interfaces. The DTMs representing these devices only differ in the availability of provided data, functions, and user interfaces. The different DTM categories reflect the physical entities, which are represented by a DTM. The defined DTM categories cannot describe rules for all existing physical devices, but are intended to provide a general guideline for common categories of physical devices.

4.2.3.2.2 Communication DTM

A Communication DTM is a special category of DTM containing the application software for specific communication hardware. For example, it may support configuration settings like baud-rate, start and stop bits, etc.

A Communication DTM shall provide Communication Channels (see 4.2.4.3) which provide the services to access the fieldbus (see Figure 5). The number of provided channels may depend on the configuration of the Communication DTM (i.e. it might occur that a DTM temporarily provides no channel, for instance during reconfiguration).

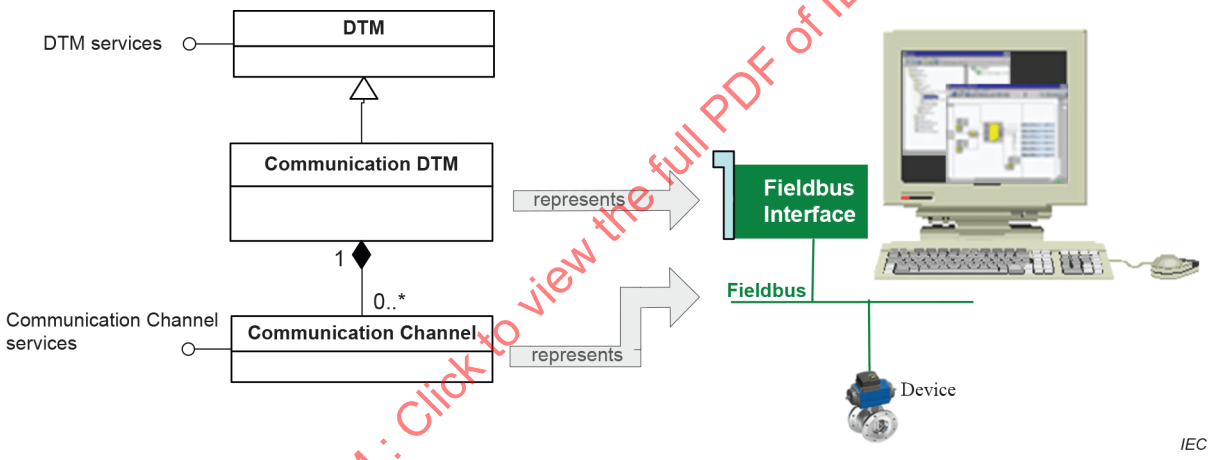


Figure 5 – Communication DTM

The advantage of Communication Channels provided by a DTM compared to the channels provided by a Frame Application (see 4.2.2) is that they may be used by different Frame Applications. Each Frame Application thus may extend the number of protocols the system is able to access. Both ways of providing Communication Channels may coexist in a Frame Application.

4.2.3.2.3 Device DTM

A Device DTM is a special category of DTM containing the application software for a typical field device, for example, a pressure transmitter or a valve. Such a DTM for example supports functions to configure and parameterize the device (see Figure 6).

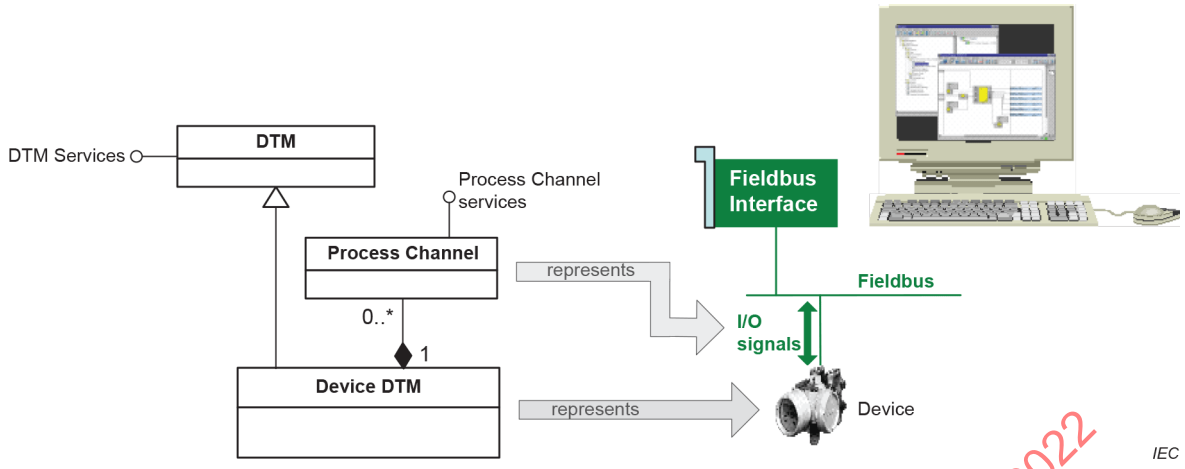


Figure 6 – Device DTM

If it is required to make the device's I/O signals or process values accessible to the host system, then the Device DTM shall provide Process Channel objects (see 4.2.4.2).

4.2.3.2.4 Gateway DTM

A Gateway DTM is a special category of DTM containing the application software for a device that connects different fieldbus segments (see Figure 7).

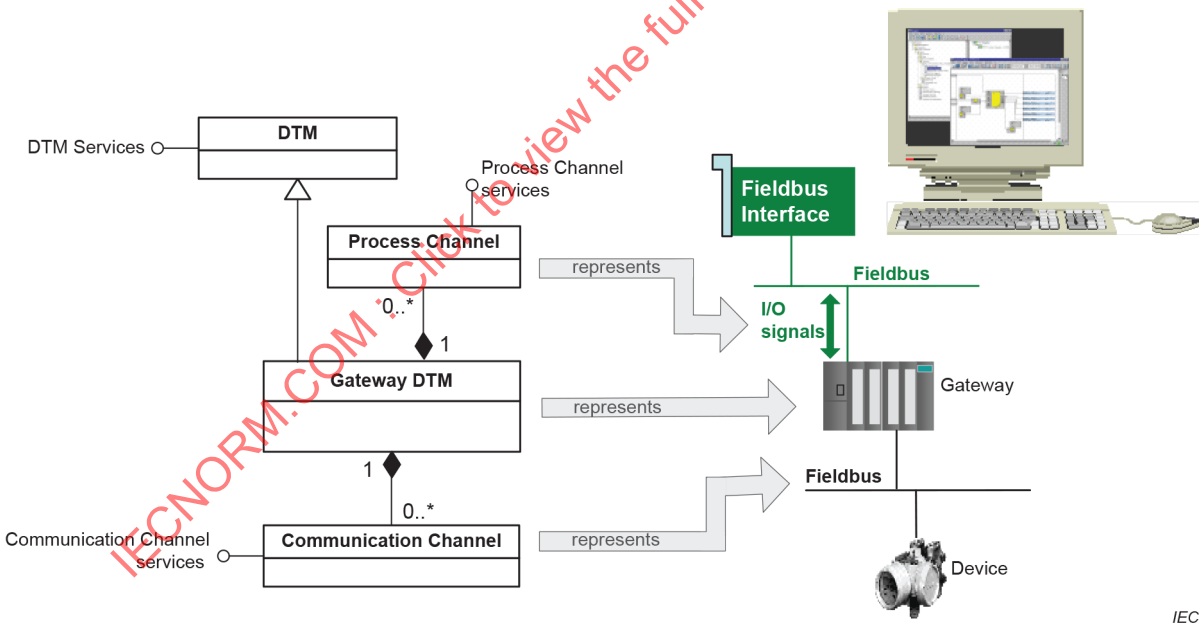


Figure 7 – Gateway DTM

Such a DTM can be seen as a combination of a Communication DTM and a Device DTM. It shall provide Communication Channels (see 4.2.4.3) to provide access to the linked fieldbus and Process Channels (see 4.2.4.2) for the fieldbus to which it is connected, if it is required to make the gateway I/O signals or process values accessible to the host system.

There may be a relation between the I/O signals provided by a gateway device and the communication interface of that device (e.g. the gateway provides I/O signals that are received from a device connected to the communication interface). In such a case, the relation is represented as a relation between the Process Channels and the Communication Channel.

A Child DTM is connected to the Gateway DTM via the Communication Channel of the Gateway DTM. The Frame Application is responsible for creating and managing the link between the DTMs. The Device DTM is able to communicate with its device via the services provided by the Communication Channel. This concept is called "Nested Communication" and is further described in 4.7.

4.2.3.2.5 Composite Device DTM

Some devices are composed of modules and sub-modules. Hardware modules are plugged into physical slots of the composite device (modular device). For example, an I/O module can be plugged into a Remote I/O device. The communication between the composite device and the modules is provided usually by a backplane bus. The function of the backplane bus is defined by the manufacturer and often is based on private protocol definitions. Such a composite device may be represented by a hierarchy of DTMs as shown in Figure 8.

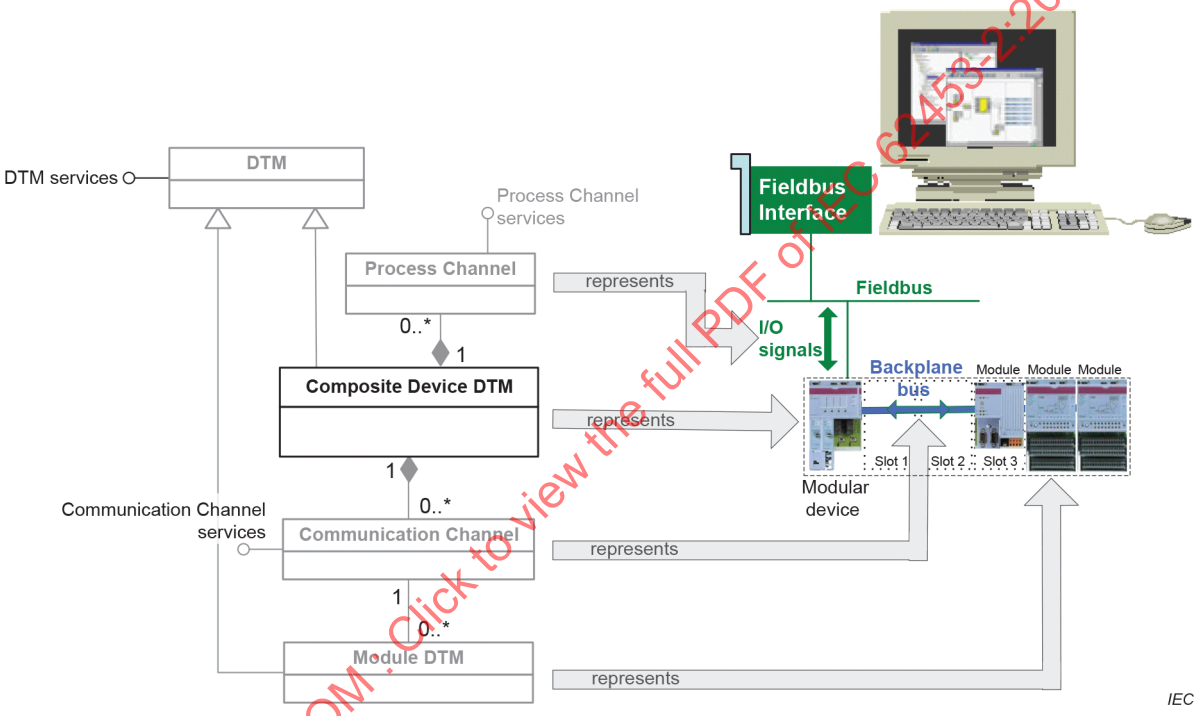


Figure 8 – Composite Device DTM

A Composite Device DTM is the root element for the complete device representation. This DTM shall provide Communication Channels (see 4.2.4.3) which represent the backplane bus. Depending on the software design of the Composite Device DTM one channel may be provided for representation of the whole backplane bus or multiple channels may be provided for representation of the slots where the modules are plugged in.

If the composite device provides process data, then the Composite Device DTM shall provide Process Channels (see 4.2.4.2). If the composite device forwards process data that is generated in the device modules, then the Process Channels provided by a Composite Device DTM may be composed of Process Channels that are provided by the Module DTMs.

4.2.3.2.6 Module DTM

A Module DTM is a special category of DTM containing the application software for a device hardware or software module (see Figure 9).

Some devices are subdivided into modules and sub-modules. A module is either a hardware module or a software module. Hardware modules are plugged into physical slots of the device. For example, an I/O module can be plugged into a Remote I/O device. Such a modular device may be represented by several DTMs as shown in Figure 9.

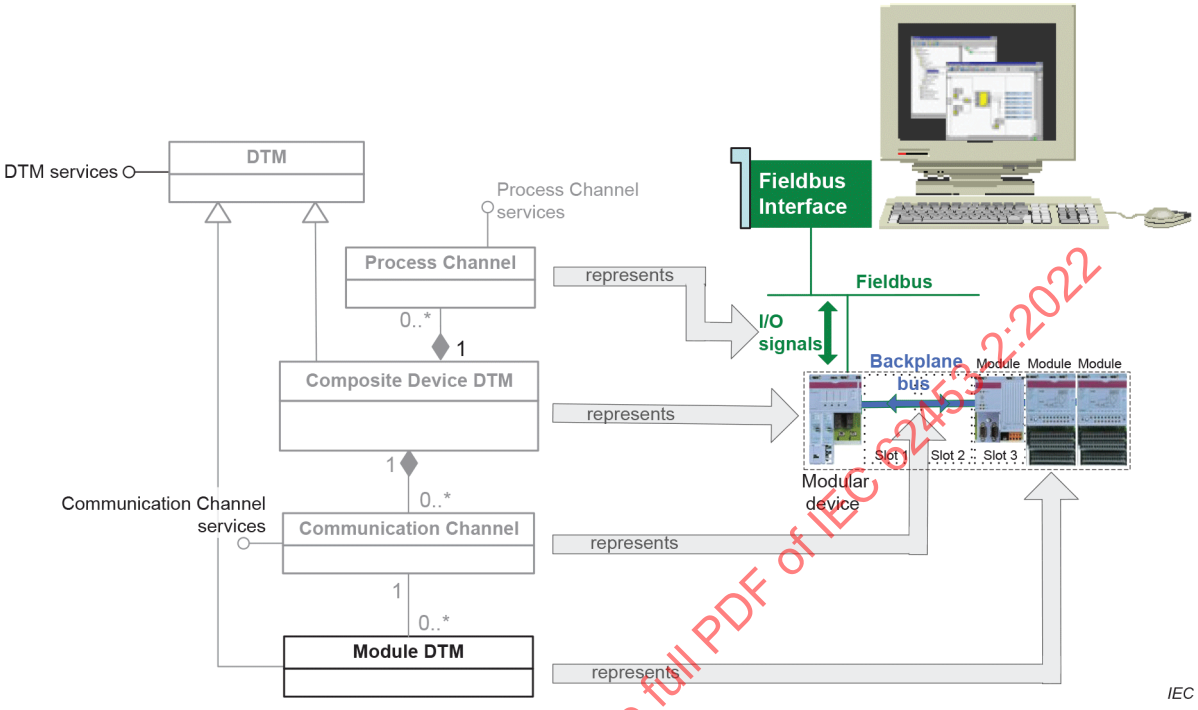


Figure 9 – Module DTM

The modules are represented by Module DTMs. The Composite Device DTM and Module DTMs are generally shipped together and supplied by the vendor of the modular device. The relation between Composite Device DTM and related Module DTMs (and their respective DtmTypes) is defined by a specific ProtocolId for the backplane bus protocol.

A Module DTM is connected to the Composite Device DTM via the Communication Channel of the Composite Device DTM. The Frame Application is responsible for creating and managing the link between the DTMs. The Module DTM is able to communicate with its module via the services provided by the Communication Channel. This means that the concept "Nested Communication" (see 4.7) is also applied for Composite Device DTMs and the related Module DTMs.

If the device modules provide process data, then the respective Module DTMs should provide Process Channels.

If the device modules provide access to subordinate communication (e.g. in a modular gateway), then the respective Module DTM shall provide its own Communication Channels.

4.2.3.2.7 Block Type Manager (BTM)

A BTM is a special category of DTM containing the application software for accessible objects inside a device. Often these software objects are defined as blocks, such as resource blocks, transducer blocks, and function blocks (see Figure 10).

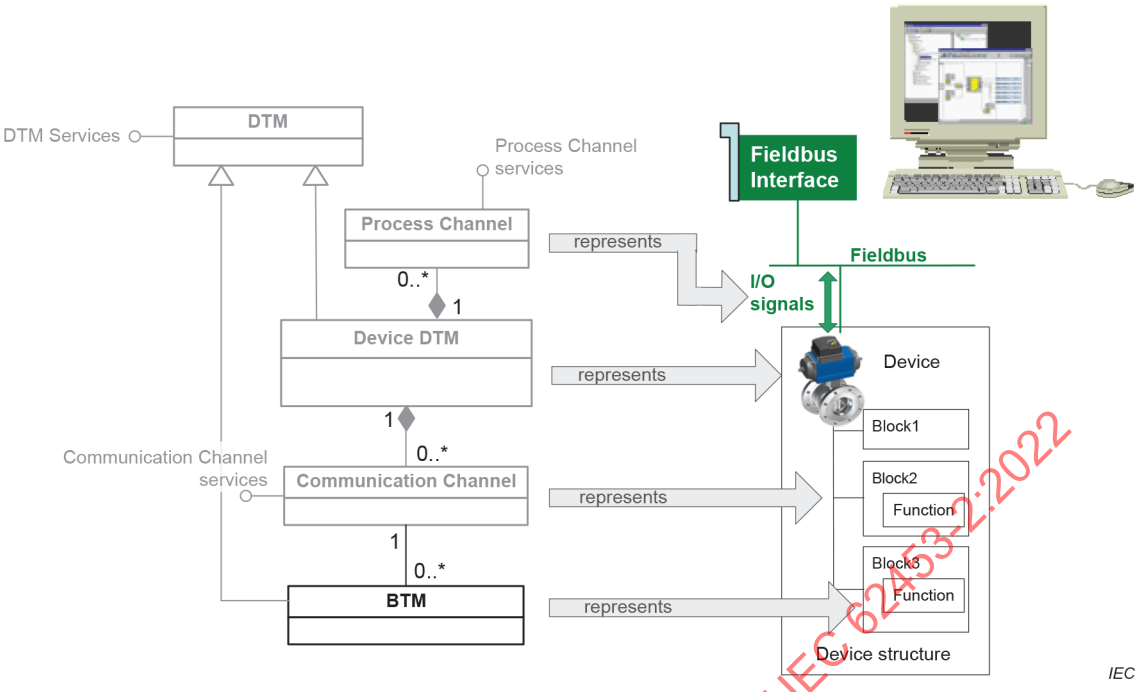


Figure 10 – Block Type Manager (BTM)

A Device DTM is the root element for complete device representation. This Device DTM shall provide Communication Channels (see 4.2.4.3) which provide services to access the block information within the device.

A device may host standard blocks, vendor-specific blocks, and device-specific blocks. All types of blocks are represented by BTMs. The software blocks are more flexible than device modules (see 4.2.3.2.6), for instance blocks may be moved between devices (standard blocks may even be moved between devices of different type). The Device DTM and the BTMs for device-specific blocks are generally shipped together and supplied by the device vendor. BTMs for standard blocks may be supplied seperately and may be used with all DTMs for devices that are able to host the respective blocks.

A BTM is connected to the hosting Device DTM via its Communication Channel, indicating that the respective blocks are hosted in the physical device. The BTM is able to interact with its block via the services provided by the Communication Channel of the Device DTM. The Communication Channel represents an internal communication on the device and is used mainly to manage the link between BTM and block. The protocols supported by the Communication Channel are used to manage which BTMs may connect to a DTM (standard protocols for standard blocks, specific protocols for specific blocks). Additional validation rules may be applied by the Communication Channel. The Frame Application is responsible for creating and managing the link between the Device DTM and the BTMs.

The BTM concept is independent from fieldbus protocol. However, a protocol may require modeling of device structures as blocks. Thus, whether and how the BTM concept is used to model device structures is defined by the IEC 62453-3xy documents describing the protocol profile integration in FDT.

If the blocks provide process data, the respective BTMs may provide Process Channels.

4.2.3.3 DTM Functions

4.2.3.3.1 Presentation object

Each of the device-specific objects (DTM, BTM and Communication Channel) may be packaged together with Presentation objects which support the services defined in 7.3 (see Figure 11).

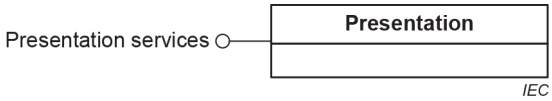


Figure 11 – Presentation object

These Presentation Objects are implementation-specific objects that provide a graphical user interface for the business object.

The Presentation Object may be a type of object, that allows integration into the GUI of the FA or it may be an object that runs outside of the GUI of the Frame Application (e.g. a standalone executable).

4.2.3.3.2 Command functions

Command functions are used to execute actions (commands) without user interface on the DTM. Such functions are executed with the service InvokeFunction (see 7.2.5.2).

One example for command functions is: Reset device.

4.2.3.3.3 Static Function

Static Function is a concept that allows processing information from a device without executing the DTM.

For observation and monitoring of a plant often it is necessary to retrieve not much data from many devices. In such use cases the effort to start and manage many DTMs may be too high. On the other hand, in such use cases it is possible that it will be not sufficient to read a single item from the device. Sometimes the data read from the device needs to be processed or to be combined with additional information in order to calculate the correct information.

A Static Function is part of a DTM package, the DTM provides information regarding the Static Function. A Static Function has no access and no relation to the data of a DTM.

Static Functions are executed using a StaticFunctionProvider object. The Frame Application is responsible for creating the StaticFunctionProvider objects. There shall be only one StaticFunctionProvider object per device and it is initialized with the necessary communication data. On one StaticFunctionProvider object the execution of a Static Function can be started multiple times in parallel. It is the responsibility of the StaticFunctionProvider to process the functions sequentially in the order in which they were started.

It is recommended that a DTM provides Static Functions to support well defined use cases as defined by FDT protocol profile integration specifications.

NOTE It is not expected that StaticFunctions provide functionality specific for a Frame Application. Instead it is expected that Profile Working groups (e.g. a working group defining a profile for support of MES functions or a working group defining a DTM profile for support of predictive maintenance) define standard functions that are used in many applications.

Examples of Static Functions are:

– GetDeviceStatus

  Provides the current device status in FDT format (see Fdt.Dtm.DeviceStatus)

– GetProcessValue

  Provides the current process value of the device (input or output) in normalized format

– GetOperationTime

  Provides the operation time of a device in standard format

4.2.4 Channel object

4.2.4.1 Channel overview

There are three different types of channels, the Communication Channels, the Process Channels and combinations of them as shown in Figure 12.

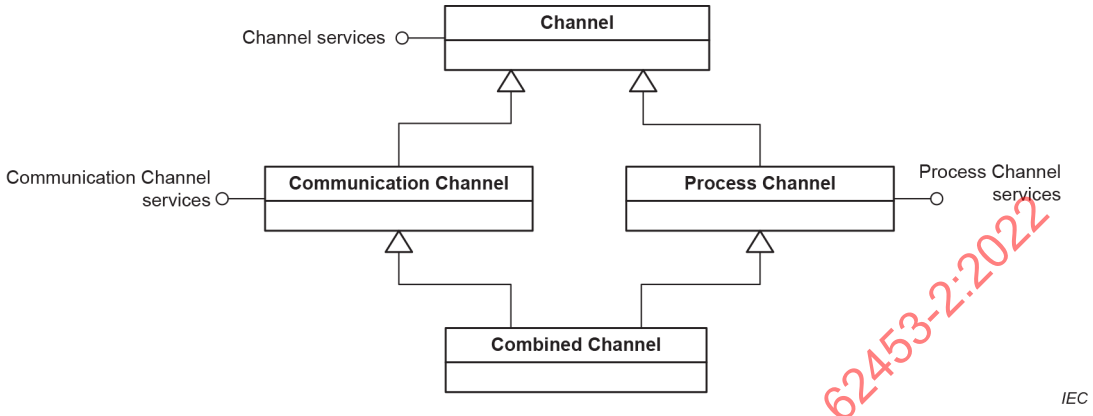


Figure 12 – Channel object

4.2.4.2 Process Channel

Input and output signals provided by devices are created and integrated into the function planning of the control system. If the device provides I/O signals the associated DTM shall offer information about the I/O signals of its device. This information is conveyed by Process Channels.

A Process Channel supports the services defined in 7.5. These services for example provide access to I/O signal information such as addressing, signal and data type, but not to the current value of the I/O signal itself.

The implementation of Process Channel and its respective services depends on the implementation technology and differs for IEC TR 62453-41 and IEC TR 62453-42.

The I/O signal information provided by the Process Channel is protocol-specific. Which information shall be provided by Process Channels is defined by the relevant IEC 62453-3xy document describing the protocol profile integration in FDT.

The number and type of I/O signals may depend on the configuration of the device. Thus the number of Process Channels and provided information may also depend on the device configuration made by the user. A Frame Application is able to inform the DTM by setting the ProtectedByChannelAssignment parameter via the service WriteChannelData (see 7.5.2) that further changes shall be prohibited, because the channel is already integrated into the functional planning of the control system (Host Channel is assigned). In this case, the DTM shall not accept any changes related to this channel.

4.2.4.3 Communication Channel

A Communication Channel provides access to the fieldbus and is communication technology dependent. It represents the entry point to a fieldbus, a vendor-specific backplane bus or a point-to-point connection. A Communication Channel may also represent logical communication with blocks within a device. A Communication Channel may be provided by a Frame Application (see 4.2.2) or by a DTM (see 4.2.3).

A Communication Channel provides communication services that are independent from the communication hardware and the fieldbus protocol. In general, the protocol-specific services are one-to-one mapped to the services provided by the channel. The interfaces provide methods to

– scan the fieldbus for available devices,
– connect to a device,
– send communication messages, and
– disconnect from a device.

The services take base arguments which are extended for protocol-specific communication by IEC 62453-3xy specifications (see Figure 13).

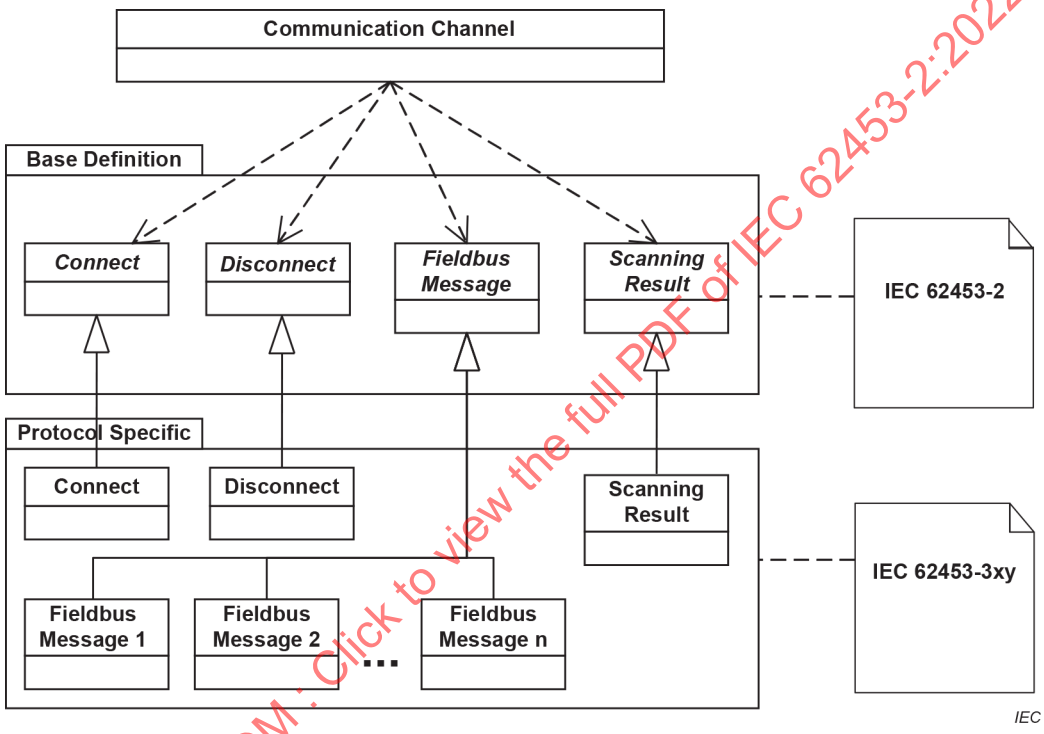


Figure 13 – Communication Channel

FDT also allows manufacturers to define support for proprietary bus protocols, which are not part of the standard, for example for a manufacturer-specific fieldbus or point-to-point communication.

The services provided by Communication Channels may be used by the Frame Application or by DTMs to exchange data with a connected device or to initiate a function (e.g. identification request, device reset, broad-cast, etc.). The general communication concept is further described in 4.7.

A Communication Channel shall be able to handle several connections at the same time if the communication protocol supports this. This means that a Communication Channel shall be able to support several DTMs creating connections to different devices as well as several DTMs creating connections to the same device. Depending on the supported protocol it may also be possible that one DTM creates several connections to one device. The Communication Channel shall guarantee that a link to a device is established as a peer-to-peer connection. The services that shall be provided by a Communication Channel are defined in 7.6. The pure communication related services (see 7.6.1) just define the frame for interaction with a connected device. The data (parameters) which shall be passed or returned by the service are defined by the IEC 62453-3xy documents describing the protocol profile integration in FDT.

In the case of vendor specific protocols (e.g. backplane bus or point-to-point) the pure communication related services (see 7.6.1) may be replaced by vendor-specific services. Following this approach only DTMs supplied by this vendor are able to communicate with each other. Therefore a vendor-specific bus category (see 4.3) shall be defined. For vendor-independent fieldbus protocols an IEC 62453-3xy document describing the protocol profile integration in FDT shall be provided and standard communication services shall be used.

4.2.4.4 Combined Process and Communication Channel

A combined Process and Communication Channel typically belongs to a Gateway DTM (see 4.2.3.2.4). It represents the I/O signal, which corresponds to the cyclic communication on the linked fieldbus and which is exposed via the fieldbus to which the gateway device is connected. At the same time such a channel represents the entry point to the linked fieldbus (or point-to-point connection) as shown in Figure 14.

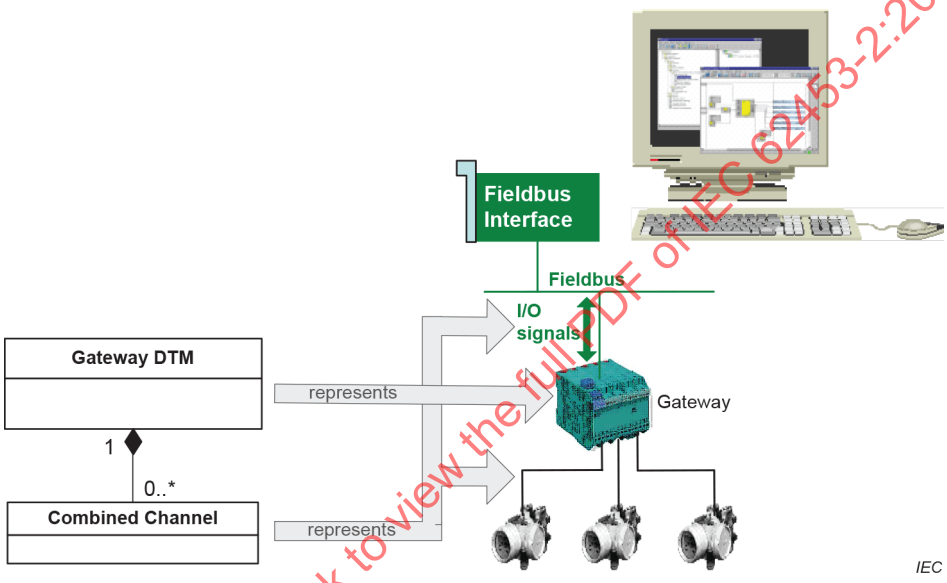


Figure 14 – Combined Process/Communication Channel

In other words, such a channel shall support the Process Channel services (see 7.5) for the fieldbus to which the gateway is connected and the Communication Channel Services (see 7.6) for accessing the devices connected to the linked fieldbus (or point-to-point connection).

4.3 Modularity

Different fieldbus protocols are using different device models. FDT supports the following different approaches to describing the structure of the device:

- monolithic DTM with DtmDeviceInstanceTopology,
- Module DTM, and
- BTM.

4.4 Bus categories

A DTM exposes information about encapsulated protocol-specific definitions. The information returned by service GetTypeInformation (see 7.2.3.1) contains so-called bus categories which are Universally Unique Identifiers for a fieldbus protocol (or a point-to-point communication).

The FDT concept distinguishes between supported and required bus categories:

- supported bus categories refer to the protocol-specific communication services provided by a DTM (corresponding Communication Channels) to access a fieldbus;
- required bus categories refer to the protocol-specific communication services required by a DTM to exchange data with its device.

4.5 Identification

4.5.1 DTM instance identification

4.5.1.1 System Tag

An FDT Frame Application shall assign a unique identifier for each DTM instance. This unique identifier is referred to as "System Tag". The System Tag is used by DTMs

– for navigation in the FDT topology,

– for the management of Child DTMs in the FDT topology (e.g. address setting).

4.5.1.2 System GUI label

The DtmSystemGuiLabel is a human-readable identifier, used to identify a DTM instance in the Project and the related DTM GUI in the display to the user. The Frame Application and the DTM should use the DtmSystemGuiLabel in order to identify dialogs and windows belonging to the DTM instance.

The DtmSystemGuiLabel may reflect an identifier of the device (e.g. the device tag) that is provided in NetworkDataInfo.Label. The DTM may use NetworkDataInfo.Label to provide an initial value for DtmSystemGuiLabel. It is recommended for a FA to use the NetworkDataInfo.Label value, when constructing the DtmSystemGuiLabel.

4.5.1.3 Hardware identification

A DTM supports the HardwareInformation service (see 7.2.3.3) that enables to read device information online from the connected device (see Figure 15).

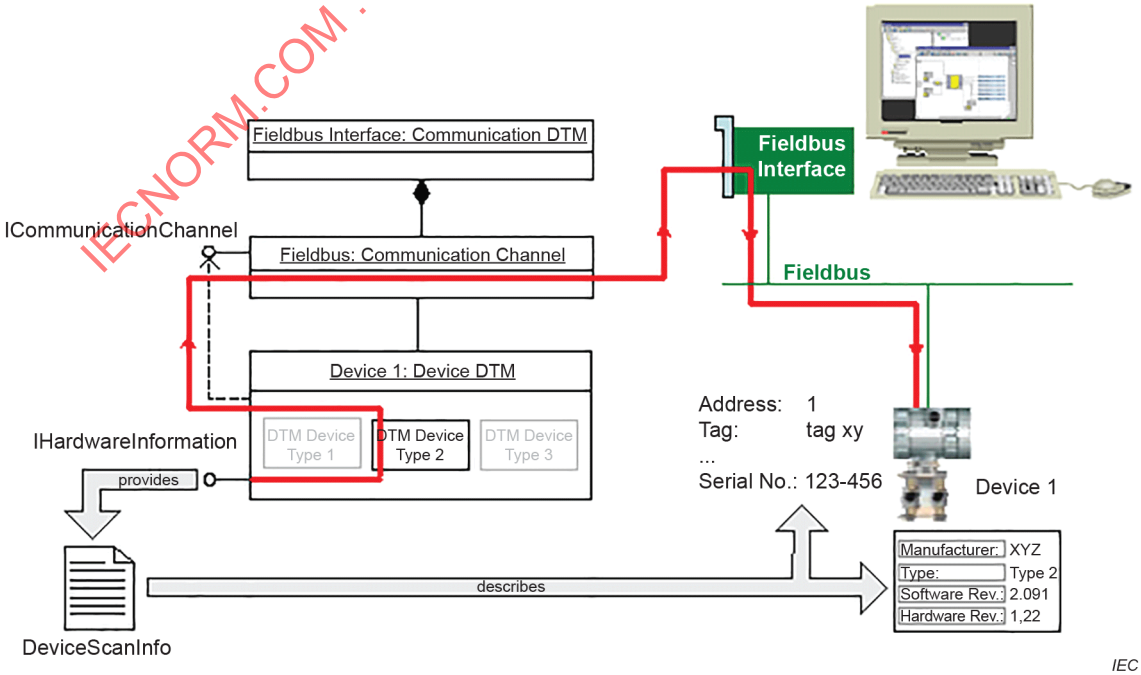


Figure 15 – Identification of connected devices

The service returns device type related information such as manufacturer, type, hardware and embedded software version as well as device instance related information such as fieldbus address, tag and serial number. This information is also fieldbus-specific like the information returned by the service GetIdentificationInformation. Therefore, the concrete protocol-specific format and transformation to the protocol-independent format which can be used by the Frame Application without fieldbus-specific knowledge are defined by the IEC 62453-3xy document describing the protocol profile integration in FDT.

4.6 System and FDT topology

The Frame Application is responsible for managing the system topology. That means the Frame Application shall organize the routing for accessing a device in the plan. A DTM shall not need to manage any information about the system topology.

System topology management is Frame Application specific. Some Frame Applications may require user interactions; others may support automatic operations such as topology importing or fieldbus scanning (see 6.2).

A DTM exposes all required information (see 7.2.3) which enables the Frame Application (and the user) to choose the appropriated DTM for a device, for example name, vendor, version of supported device types and corresponding identification properties.

The Frame Application initializes and links the DTMs according to the system topology. The linking of DTMs is called FDT topology. Figure 16 shows the FDT topology for a simple system topology with one fieldbus and two connected devices.

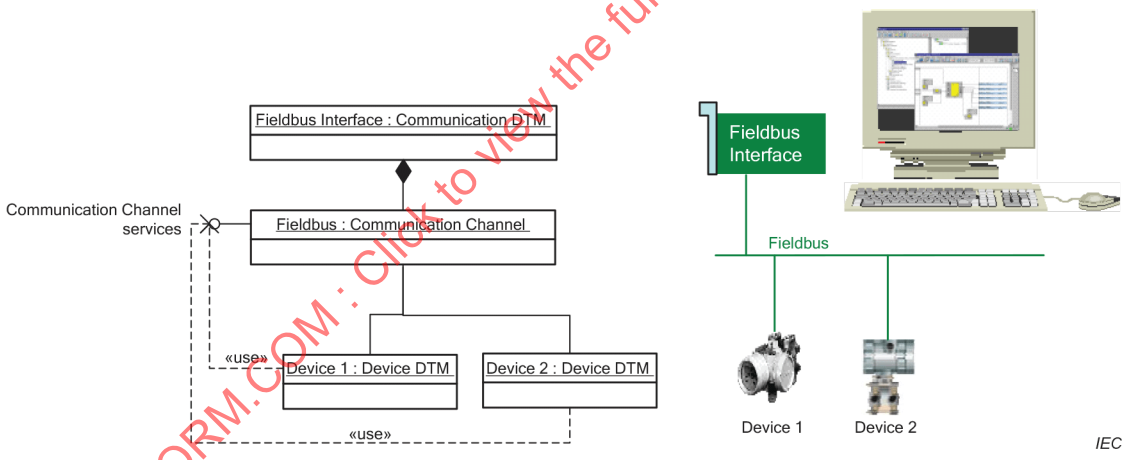


Figure 16 – FDT topology for a simple system topology

A Communication Channel is always used as the linking element between DTMs. As shown in Figure 16, the two Device DTMs are linked to the Communication Channel which provides access to the fieldbus.

The link between a Communication Channel and a DTM is created by the Frame Application. However, final decision whether a DTM shall be linked or not has to be made by the Communication Channel. The Frame Application has to call the service ValidateAddChild (see 7.6.2.1) before the link is created. The Communication Channel shall at least check whether the required bus categories of the DTM to be linked fit its own supported bus categories. If this is not the case, then linking shall be rejected. In addition, the Communication Channel may perform additional checks, for example if the number of linked DTMs exceeds a limit. The sequence diagrams in 8.1 describe this sequence in more detail.

Neither the Communication Channel (or corresponding DTM) nor the linked DTM shall need to manage topology information. The Frame Application supports requesting topology information by service GetParentNodes (see 7.7.3.5) and service GetChildNodes (see 7.7.3.6).

The FDT topology management for a more complex system topology with gateways, modular devices and devices with blocks works exactly the same way.

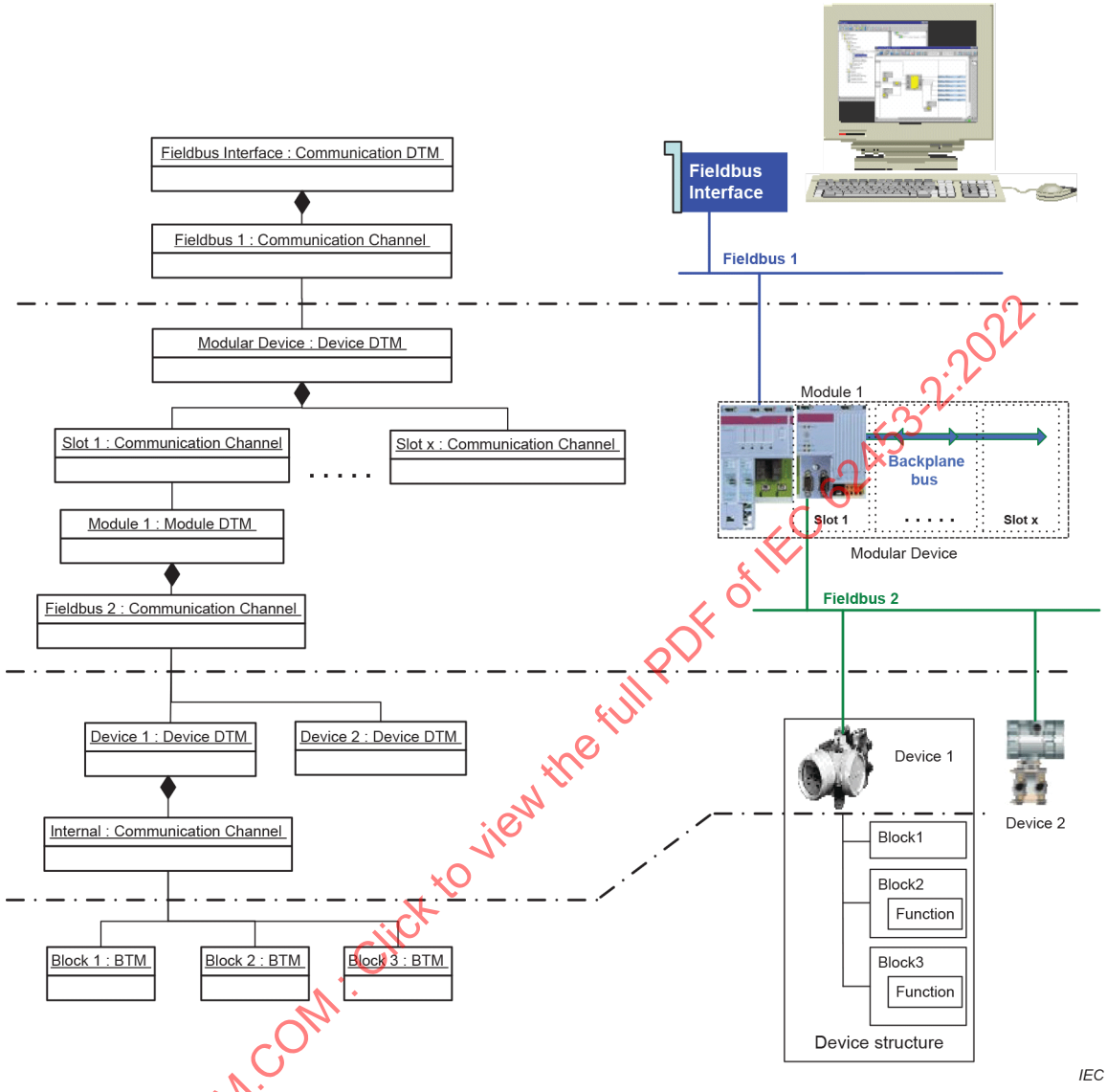


Figure 17 – FDT topology for a complex system topology

Starting from the root, all DTMs are linked via Communication Channels according to the physical system topology.

The root element DTM for a device which is represented by multiple DTMs (e.g. Modular Device and Device 1 DTMs in Figure 17) may support the automatic creation of the FDT sub-topology that corresponds to the complete device structure. The Frame Application supports the service CreateChild (see 7.7.3.2), DeleteChild (see 7.7.3.3) and MoveChild (see 7.7.3.4) to enable a DTM to alter the FDT topology.

4.7 FDT Communication

4.7.1 General

In FDT the communication is based on passing communication requests from the consumer of communication to the communication provider, which is represented by a Communication Channel. Passing and receiving communication requests is based on asynchronous services (see 8.3).

In order to submit communication requests the consumer of communication first has to establish one or more connections to the respective physical device. This allows the management of active connections by the Communication Channel.

4.7.2 Handling of communication requests

The Communication Channel receives one or multiple transaction requests from the consumer of communication and executes these requests. In general, the Communication Channel is expected to process the transaction requests in the order in which they are provided. The results of the transaction requests are passed back to the client of the Communication Channel.

The relation between communication requests and communication responses may be managed by an ID, this is implementation specific. For example, each transaction request can be identified by an ID, and the same ID is provided in the transaction response.

4.7.3 Handling of communication errors

The communication client shall evaluate each of the received transaction results in order to recognise possible communication errors.

If a communication client receives errors for requests that originated at that client, then the client shall inform the user. In a scenario with nested communication this means the Gateway DTMs shall not inform the user about communication errors for requests that originally came from a Child DTM. The Child DTM has the responsibility for informing the user.

4.7.4 Handling of loss of connection

If the connection to a device is lost irreparably the Communication Channel shall abort the respective connection(s) and send an Abort notification to the respective communication client for each aborted connection (see 7.6.1.5). The AbortMessage provides the CommunicationReference for the aborted connection. If a DTM creates multiple connections to the device, each connection is managed separately. The abort of each connection may depend on specific conditions depending on the used communication protocol.

The specific triggers for sending the Abort notification (e.g. device does not respond) will be defined specifically for each communication protocol.

If a Gateway DTM receives an Abort notification, then it shall send Abort notifications to all connected communication clients.

The origin of the Abort notification shall not provide a user message to inform the user about loss of connection. The communication client receiving the Abort notification shall inform the user about the loss of connection, e.g. if the communication client has open user interfaces, then the user shall be informed about the loss of connection by updating the connection status in the user interface. If the communication client is composed of several DTMs which may have open connections at the same time (e.g. a Composite Device DTM with several Module DTMs), then the communication client should avoid providing several user messages, but should inform the user about loss of connection with one message.

After sending an Abort notification the Communication Channel shall not send any further transaction responses to the communication client. The communication client should ignore any transaction response received after receiving an Abort notification.

4.7.5 Point-to-point communication

The Frame Application has to provide in each case a peer-to-peer connection between a DTM and a device.

It is under the control of the Frame Application to enable a DTM to communicate with its device. The Frame Application has to pass the Communication Channel to use the DTM by calling the service SetLinkedCommunicationChannel (see 7.2.4.2) and allow the Communication by calling the service EnabledCommunication with TRUE (see 7.2.4.3).

In order to access the device, the DTM uses the Communication Channel services (see 7.6.1) as shown in Figure 18.

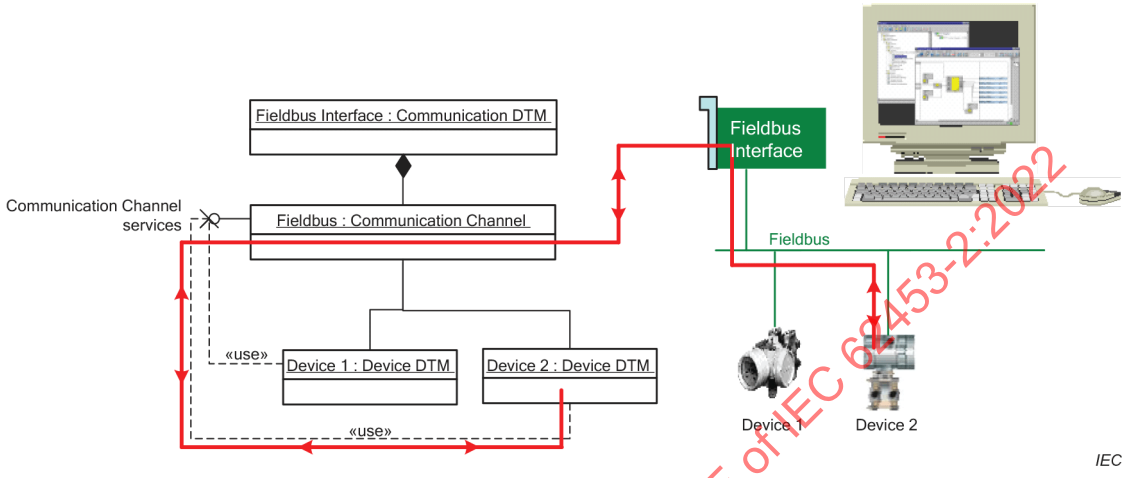


Figure 18 – Point-to-point communication

A DTM has to call the Communication Channel service Connect (see 7.6.1.2) to establish a communication link to the device. After the link is established, it is able to communicate with its device by calling the service Transaction (see 7.6.1.6). The diagram in 8.3.2 describes this sequence in more detail.

DTMs shall consider the limited availability of fieldbus resources when connecting to the device. This means a DTM should create only as many connections as needed and connections shall be terminated if they are not used (e.g. the online function is finished).

4.7.6 Nested communication

A DTM shall not need to know anything about the system topology in order to access the corresponding physical device. Thus the same DTM is able to communicate with devices attached to different fieldbus interfaces of the same protocol. Nested communication is used if the devices are connected to a sub-system, for example if a device is connected to a channel of a Remote I/O (see Figure 19).

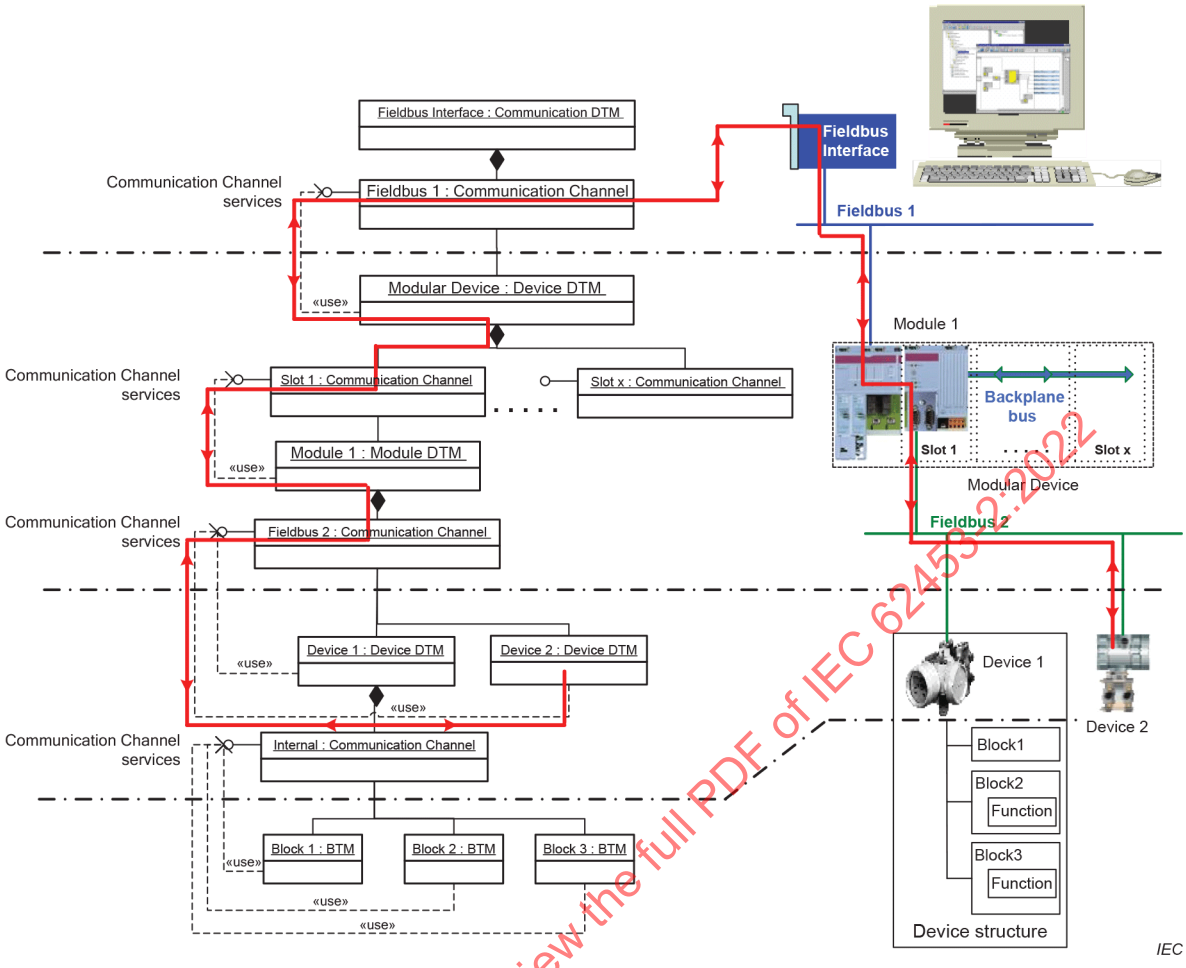


Figure 19 – Nested communication

Starting from the root, the Frame Application shall manage the link between Communication Channels and the respective DTMs, which should be enabled for communication with their respective devices. As in the point-to-point communication, all DTMs simply use the Communication Channel services to communicate with their devices without the awareness of the nested communication (e.g. device 2 DTM in Figure 20). The diagram in 8.3.3 describes this sequence in more detail.

The communication in FDT follows the concept of nested communication:

- each Communication Channel wraps the communication message from the linked DTM below, without knowing the contents;
- the linked DTM below the Communication Channel does not have any knowledge about the system topology;
- the DTM shall support only communication protocols that are supported by the respective device;
- communication/routing through any system topology is possible without limitations.

See 8.3.3 for a communication sequence related to nested communication.

4.8 DTM, DTM Device Type and hardware identification information

4.8.1 DTM and DTM Device Type

A DTM supports two services to request information about itself and the supported device types (see Figure 20). Usually, this information is used by the Frame Application to create libraries, select a DTM during creation of FDT topology, or for simple validations.

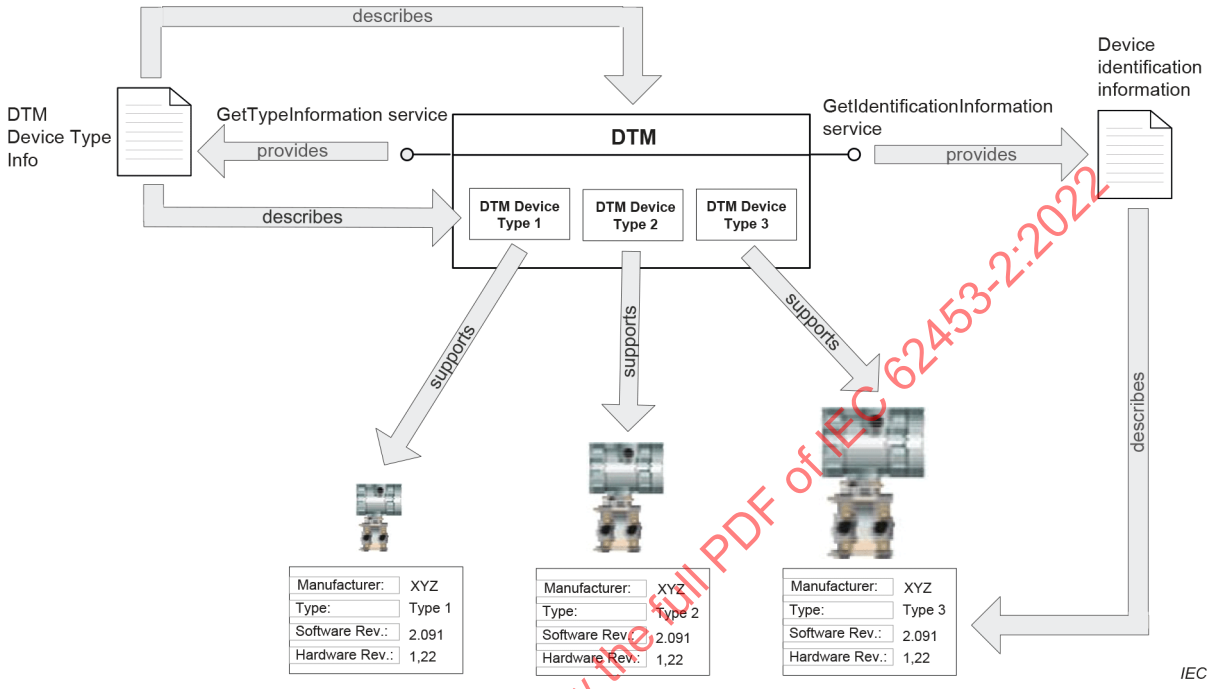


Figure 20 – DTM, DTM Device Type and device identification information

A DTM is a software component containing device-specific application software. The service GetTypeInformation (see 7.2.3.1) returns general information about this piece of software such as name, version, and vendor of the software, the FDT version (see also 4.13) as well as a list of supported device types.

The representation for a particular device type within the DTM is called DTM Device Type. A DTM may contain one or more DTM Device Types. The concrete design and implementation of the DTM Device Types is not within the scope of the FDT specification; but the service GetTypeInformation also returns information about these pieces of software such as name, version, vendor, supported protocols, etc.

When a Frame Application adds a DTM to the FDT topology then it selects one of the supported DTM Device Types (see service Initialize, 7.2.4.1). The DTM shall persist the selection within its instance data, in order to restore it when a DTM representing the same device is initiated the next time. The DTM shall provide information about the selected DTM Device Type with the service GetActiveTypeInfo (see 7.2.3.4). The service shall always return the current DTM Device Type. If an update of the DTM occurred, the new DTM shall return the updated DTM Device Type information.

4.8.2 Supported hardware identification

Information about physical device types, which can be handled by the DTM Device Types is returned by the service GetIdentificationInformation (see 7.2.3.2), for example the manufacturer, type, hardware and embedded software version of the device. The DTM may even return wildcards for some specific device identification elements to signal that the DTM Device Type can be used for all devices for which the wildcard matches (e.g. the character asterisk '*' for the hardware version may signal that the DTM Device Type supports all hardware versions).

The information returned by the service GetIdentificationInformation is fieldbus-specific and therefore defined by the IEC 62453-3xy document describing the protocol profile integration in FDT. However, FDT defines the means to transform the information into a protocol-independent format to enable Frame Applications without protocol-specific knowledge to use it. The transformation is implementation technology dependent and therefore defined in the object model integration profile documents of the IEC 62453 series describing the mapping to specific implementation technologies.

4.8.3 Connected hardware identification

Furthermore a DTM supports service HardwareInformation (see 7.2.3.3) that enables to read device information online from the connected device (see Figure 21).

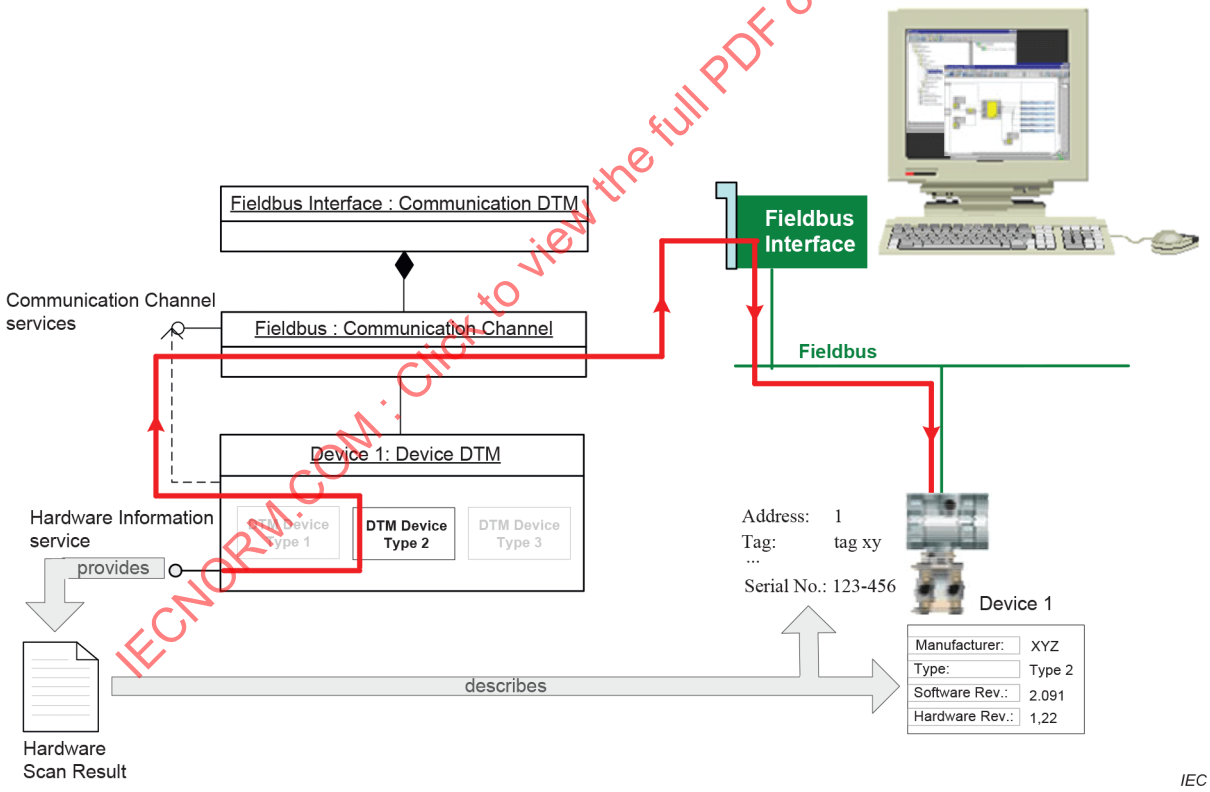


Figure 21 – Connected hardware identification

The service returns device type related information such as the manufacturer, type, hardware and embedded software version as well as device instance related information such as fieldbus address, tag and serial number. This information is also fieldbus-specific such as the information returned by GetIdentificationInformation (see 7.2.3.2). Therefore, the concrete format and transformation to protocol-independent format is also defined by the IEC 62453-3xy document describing the protocol profile integration in FDT. See examples in 6.2.4 and 6.2.5.

4.9 DTM data persistence and synchronization

Basically, three kinds of DTM related data can be seen:

- instance-related data (called "instance data"). Instance-related data belongs to the DTM itself. It is up to the DTM which data it stores, but the DTM has to guarantee that it is able to represent the stored device instance by loading this data;
- non-instance-related data. Non-instance-related data belongs to a Project or is global or DTM Device Type specific;
- bulk data. DTM specific data, for example historical data, temporary data.

The format of all kinds of data is DTM-specific, but a DTM shall expose a Universal Unique identifier specifying the format which is used to save its instance data. Furthermore, a DTM shall provide a list of compatible data formats it is able to load. These format identifiers are included in the DTM Device Type information returned by GetTypeInformation (see 7.2.3.1). A Frame Application may use the format identifiers to find the correct DTM after a DTM software upgrade (see 8.9).

Two types of DTM data persistence mechanism are defined in FDT. A DTM shall either use the Frame Application Project storage or a private DTM storage (see Figure 22).

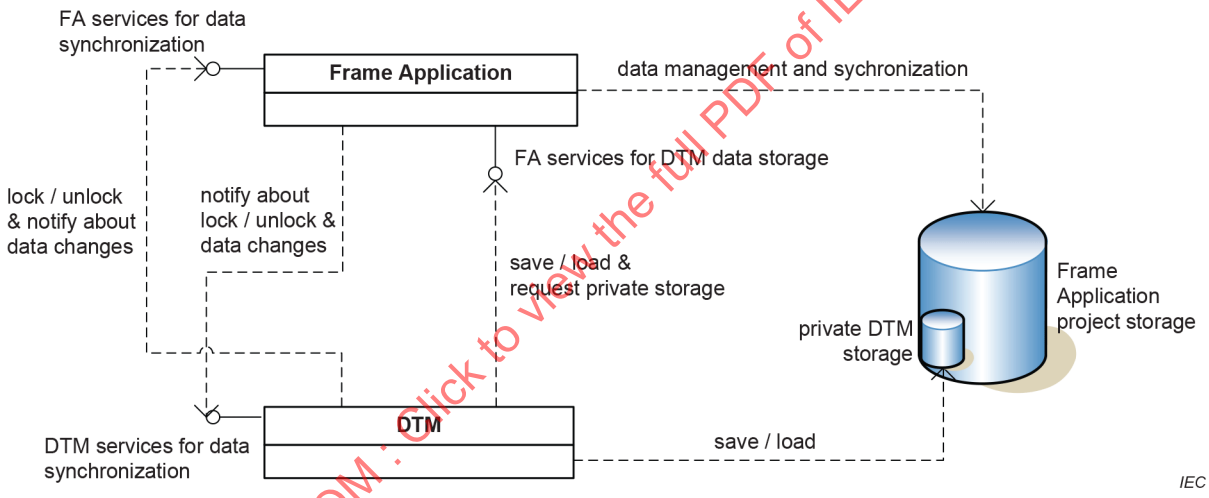


Figure 22 – FDT storage and synchronization mechanisms

The service LoadInstanceData (see 7.7.5.1) and service SaveInstanceData (see 7.7.5.2) enable the DTM to save and load instance-related data in the Frame Application Project storage. The Frame Application has to guarantee the data consistency for multi-user and multi-client data access. The 'FA services related to DTM data synchronization' (see 7.7.6) shall be used by the DTM to attempt locking of its instance-related data before alternation. The 'DTM services related to data synchronization' (see 7.2.13) shall be used by the Frame Application to notify a DTM about events, for example if data was locked by another DTM instance. See 8.5.

The service GetPrivateDtmStorageInformation (see 7.7.5.3) returns information about private DTM storage which is directly accessible for the DTM without the need to use further services provided by the Frame Application. The private DTM storage enables the DTM to store instance-related, non-instance-related, and bulk data. The DTM itself has to guarantee the data consistence for multi-user and multi-client data access. It is the responsibility of the Frame Application to create a backup strategy for the private DTM storages.

A DTM shall be able to work without any side effects if the private DTM storage is not available. It shall ensure that there is no impact to the instance data managed by the services SaveInstanceData (see 7.7.5.1) and LoadInstanceData (see 7.7.5.2).

The private DTM storage mechanism is implementation technology dependent and thus defined in the object model integration profile documents of the IEC 62453 series defining mapping to specific technologies.

4.10 DTM device parameter access

A DTM supports services to read and write device parameters stored in the DTM instance data (see 7.2.8) and directly in the connected device (see 7.2.10).

It is specific to a DTM and depends on the device- and fieldbus-type which device parameters are available. Semantic information (see SemanticId in Table A.4) is given to the device parameters. By using these, for example a Frame Application is able to get the information regarding the meaning and usage of a single parameter or data structure. The definition regarding the identifier is protocol-specific and thus defined in the IEC 62453-3xy specifications defining the protocol profile integration in FDT.

4.11 DTM state machine

4.11.1 DTM states

The following diagram (see Figure 23) defines the different states of a DTM.

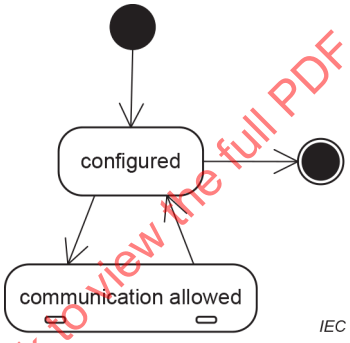


Figure 23 – DTM state machine

The state machine consists of two states: 'configured' and 'communication allowed'. Which DTM services are available in a certain state is defined in the service descriptions (see Clause 7).

All state transitions are triggered by the Frame Application by calling corresponding DTM services. The state transition succeeds if the service returns fdt:serviceSucceeded = TRUE. The DTM remains in its previous state if state transition is rejected by returning fdt:serviceSucceeded = FALSE. The circumstances under which the DTM is allowed to reject the state transitions are defined in the service descriptions.

State Transition: Initial state – state 'configured'

The service Initialize (see 7.2.4.1) shall be called to bring the DTM from its initial state into the state 'configured'.

State Transition: State 'configured' – state 'communication allowed'

The service EnableCommunication (see 7.2.4.3) shall be called with TRUE to bring the DTM from the state 'configured' into the state 'communication allowed'. A precondition for this call is that the Frame Application has informed the DTM about the communication infrastructure by calling the service SetLinkedCommunicationChannel (see 7.2.4.2).

Only in this state the DTM is allowed to establish a communication link to its device by calling the service Connect (see 7.6.1.2) of its linked Communication Channel. The substates within this state are described in the communication state machine (see 4.11.2).

State Transition: State 'communication allowed' – state 'configured'

The service EnableCommunication (see 7.2.4.3) shall be called with FALSE to bring the DTM from the state 'communication allowed' into the state 'configured'.

State Transition: State 'configured' – final state

The service Terminate (see 7.2.4.6) shall be called to bring the DTM from state 'configured' into its final state.

Table 3 shows an overview of these transitions.

Table 3 – Transitions of DTM states

Start state	End state	Trigger	Condition
Initial state	Configured	Initialize	
Configured	Communication allowed	EnableCommunication(TRUE)	Communication infrastructure available to DTM
Communication allowed	Configured	EnableCommunication(FALSE)	
Configured	Final state	Terminate	

4.11.2 'Communication allowed' substates

This state machine describes the substates in the DTM state 'communication allowed' (see Figure 24).

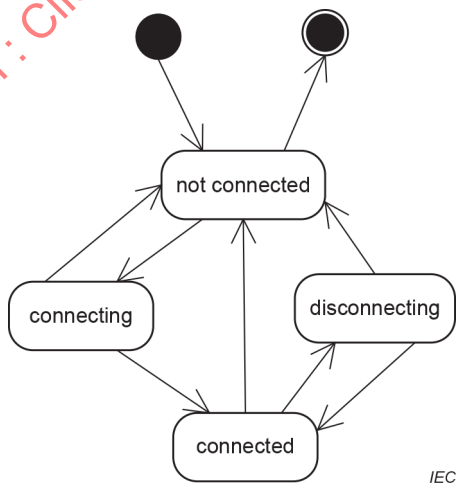


Figure 24 – Substates of communication allowed

Table 4 provides a description of the transitions of the DTM 'communication allowed' substates.

Table 4 – Transitions of DTM 'communication allowed' substates

Start state	End state	Trigger	Condition	Action
'not connected'	'connecting'	Trigger of Online service of the DTM		Connect Request
'connecting'	'not connected'		Connection could not be established	
'connecting'	'connected'	Connect Response		
'connected'	'not connected'	Connection is terminated by communication		
'connected'	'disconnecting'	Online service of DTM is finished		Disconnect service
'disconnecting'	'connected'		Connection could not be terminated	
'disconnecting'	'disconnected'	Disconnect Response		

4.12 Basic operation phases

4.12.1 Roles and access rights

When a DTM is started by the Frame Application, the user role is passed to the DTM, which shall restrict the parameter access according to the role.

Also the availability of functions and appearance of user interfaces may be adapted.

Examples:

- an observer will not get access to calibration functions;
- an observer shall not modify parameters.

See 5.2.

4.12.2 Operation phases

If a Frame Application requests available functions from the DTM, it passes the operation phase, which shall be used by a DTM to adapt the availability of functions and the appearance of the user interface.

There are five operation phases:

- Engineering
  - planning and configuring of a plant,
  - no online access to the plant;
- Commissioning, workshop
  - installing the plant infrastructure and the devices,
  - scanning the network to verify a planned network,
  - downloading project data into the plant,
  - programming, diagnosis of the devices,
  - adjusting parameters;

- Runtime
  - the plant is completely commissioned and running,
  - very restricted access to configuration and parameterization data,
  - scan to verify the network,
  - replacing defect devices,
  - reading process values and diagnosis information;
- Service
  - this operation phase is used for service tool Frame Applications. Scan to create a topology. Scan to verify a network. All service related operations. Unrestricted access to the device;
- Not supported
  - the application does not support the operation phases defined above,
  - a DTM shall offer all functions if the Frame Application passes "not supported".

Table 5 describes the usage of operation phases in different Frame Application types.

Table 5 – Operation phases

System Frame Application	Service tool Frame Application	Other Frame Applications
Engineering	Service	notSupported
Commissioning		
Runtime		

For example the DTM may allow the maintenance actor during the commissioning phase access to Online Parameterization for the complete parameter set, but during the runtime phase it may prevent it or it may allow Online Parameterization only for some of its parameters.

4.13 FDT version interoperability

4.13.1 Version interoperability overview

FDT is a component based concept, which is constantly enhanced to new improved versions. In contrast, control system environments typically run for 10 years to 15 years. If hardware and software components in a control system have to be exchanged, a mixture of components designed for different FDT versions may emerge.

This raises the question of how components based on different FDT versions can cooperate.

Subclause 4.13 mainly deals with two topics concerning FDT version interoperability:

a) Persistence

New versions of FDT components (Frame Applications and DTMs) shall be able to load instance data of older versions.

b) Component Interoperability

Components of different FDT versions (within one implementation technology) shall interoperate properly. DTMs of newer FDT versions shall run in older Frame Applications and vice versa. Communication shall work even if FDT versions of Device DTMs and Communication DTMs differ. Interaction between these components is technology dependent and therefore details are defined within object model integration profile documents.

A limiting condition for future FDT enhancements is to ensure maximum compatibility of different FDT versions. This results in a maximum interoperability of FDT components originally designed for different FDT versions.

Within object model integration profile documents of the IEC 62453 series, version interoperability on two different levels is considered:

1) Specification level

   The FDT specification targets full compatibility of different versions of the standard. Properly designed FDT components will achieve compatibility without extra implementations.

2) Implementation level

   Within object model integration profile documents the FDT version is composed of a major, a minor, a release and a build number (e.g. 1.2.0.3 where 1 is the major, 2 is the minor number, 0 the release and 3 the build number).

   Compatibility is defined slightly differently with respect to the major number:

   – compatibility between FDT components with the same FDT major version number is assured by the object model integration profile specification;

   – compatibility between FDT components with different FDT minor, release and build version numbers can be achieved by extra code paths. FDT specification will provide needed information and mechanisms to support full compatibility at this level. The minor or release number is increased if new functionality, dependent of its appropriate importance, is added to the FDT specification. The build number is increased if a bugfix within the specification or the type library is necessary.

4.13.2 DTM and device versions

When providing a new DTM for a new major FDT version, it is highly recommended to use new registration information (see 7.2.2). If the registration information of a DTM has changed, object model integration profile documents of the IEC 62453 series will provide a mechanism to upgrade to the newer.

Within its device catalog, a new DTM version is able to provide the same list or a subset of the devices of the old DTM version. In this case, an FDT Frame Application will handle the two versions of a DTM as any other DTMs able to handle the same field device as two distinguished DTMs.

A DTM of a higher FDT version which does not use new registration information shall support at least all DTM Device Types that were supported by previous versions of this DTM.

4.13.3 Persistence

If the version of the Frame Application changes or if the version of the DTM changes, then persistence (loading of stored Projects) may be a problem.

A Frame Application shall be able to load old instance data and to provide the unchanged instance data to the DTM even if the DTM version has changed. A DTM shall be able to load old instance data as long as the registration information of the DTM does not change. If registration information of a DTM has been changed, FDT specification provides a technology-dependent mechanism to upgrade to the newer DTM defined within object model integration profile documents. In this way, it is possible to load an existing old instance data of a DTM into another DTM object. The new DTM is responsible to convert the instance data accordingly.

A new DTM shall provide information on what DTM instance data it can load (compatibility of dataset). It shall support all DTM Device Types of the old DTM. If the Frame Application recognizes the new DTM, it may inform the user (e.g. "A new compatible DTM was installed, do you want to use the new version instead of the old version?"). If the user confirms that the new DTM object is instantiated instead of the old, it loads and converts the old instance data.

4.13.4 Nested communication

4.13.4.1 Nested communication overview

Since DTMs may be chained with respect to the FDT topology, they need to communicate properly. Because the involved DTMs may support different FDT versions, the following restrictions in 4.13.4.2 to 4.13.4.4 shall be considered.

4.13.4.2 Information exchange

Information exchanged via communication services may contain version-dependent parameters.

Therefore, a DTM shall only use communication service parameters according to the FDT version of its parent component. This version information of the parent component shall be provided within the service Connect (see 7.6.1.2) response information. A DTM shall then use only communication parameters compatible with this FDT version.

4.13.4.3 Communication Channel upgrade

A Communication Channel shall support the older FDT minor versions if it is upgraded to a higher minor version. This is due to the fact that some of its already existing children might not support the higher minor version interfaces.

4.13.4.4 Version compatibility

Within nested communication the version number returned by a Gateway DTM depends on the functionality of its parent component.

If the Gateway DTM is at a higher version number than the parent component then the Gateway DTM:

- shall return the same version as the parent and provide the functionality according to this version, or
- shall emulate the higher functionality if possible. Then the Gateway DTM shall guarantee that all required functionality is provided by its parent.

5 FDT session model and use cases

5.1 Session model overview

Clause 5 describes the use cases that were considered when designing the FDT specification. A Frame Application may support only a subset of these use cases. Also a Frame Application may support additional use cases that are not defined in Clause 5.

Figure 25 shows the main use cases.

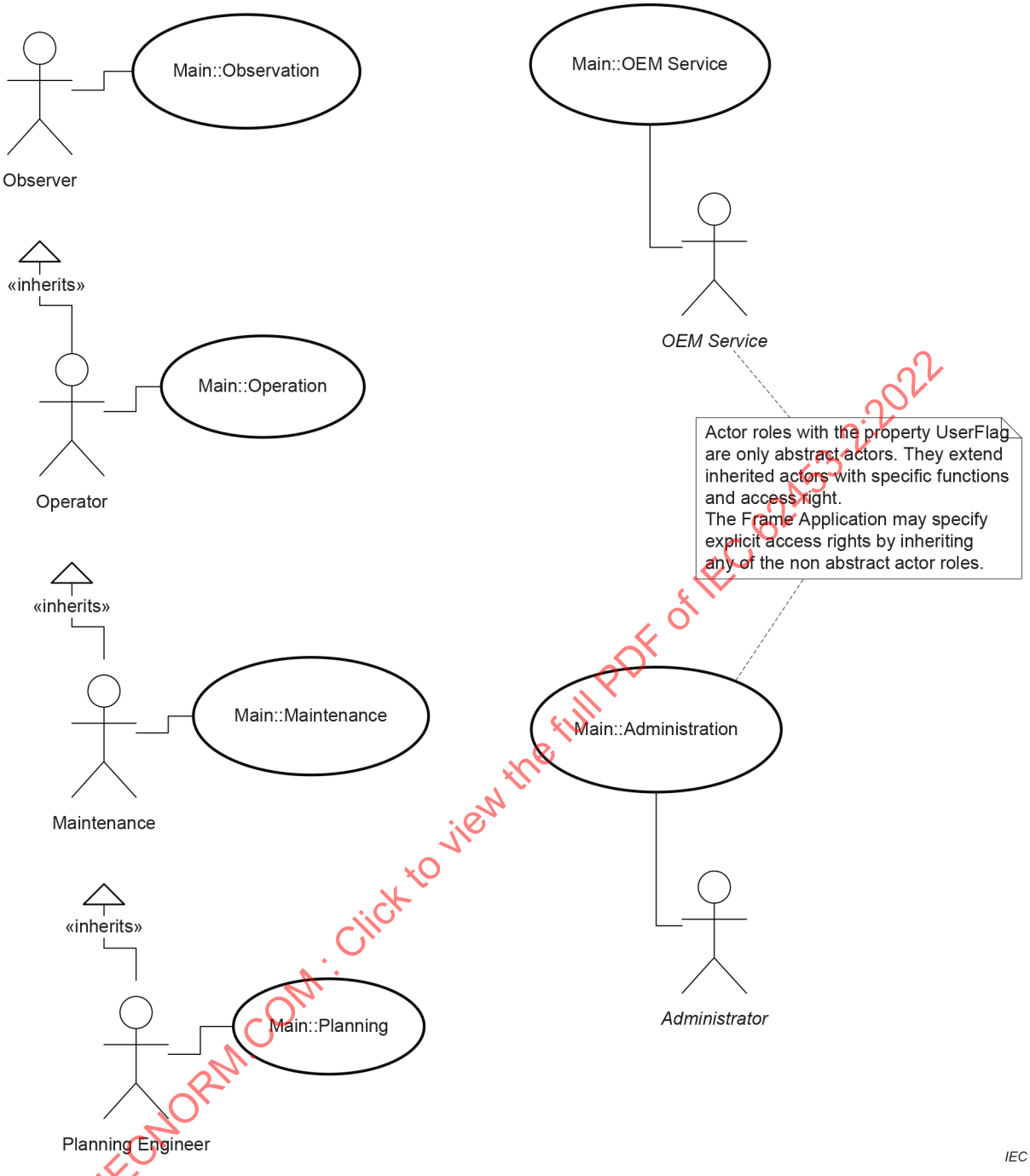


Figure 25 – Main use case diagram

The main use cases are specified in more detail in 5.3, including textual descriptions and optionally some necessary scenarios.

5.2 Actors

The actor types in the use case diagram of Figure 25 are structured in a hierarchical way.

The "Maintenance" actor inherits the "Operator" actor. The "Planning Engineer" actor inherits the use cases of the "Maintenance" actor. Both the "Administrator" and the "OEM Service" can extend the inherited actors by additional use cases. Use cases, which are passed on to an actor of higher level, may act in an extended way to match their intention.

Table 6 provides a brief description of the actors' roles:

Table 6 – Actors

Actor name:	Observer
Brief description:	This actor stands for a person that may only observe the current process
Inherits:	None
User level:	FDTUserInformation.userLevel = observer
Access verification:	The access as "Observer" actor might not have a password
Use cases:	None
Actor name:	**Operator**
Brief description:	This actor stands for a person who has to observe and manage the current process. The "Operator" may therefore check the current status of the device, modify set values and check if the device is functioning well. The use cases for this actor role should enable the user to perform a complete diagnosis, watch the actual status and parameter set as well as the current process variables
Inherits:	Observer
User level:	FDTUserInformation.userLevel = operator
Access verification:	The access as "Operator" requires a password
Use cases:	User Login, Online View, Audit Trail, Archive, Report Generation, Asset Management
Actor name:	**Maintenance**
Brief description:	A "Maintenance" person should have the possibility to perform all necessary maintenance operations including device exchange, teaching, calibration, adjustment, etc. The person may therefore download verified parameter sets, modify a subset of parameters online or offline, perform device-specific online operations and at the end of the processing, may have the possibility to upload the complete parameter set
Inherits:	Operator
User level:	FDTUserInformation.userLevel = maintenance
Access verification:	The access as "Maintenance" actor requires a password
Use cases:	Simulation, Online Operation, Repair, Offline Operation, Bulk Data Handling User Login[a], Online View [a], Audit Trail [a], Archive [a], Report Generation [a], Asset Management [a]
Actor name:	**Planning Engineer**
Brief description:	The actor "Planning Engineer" stands for person like a plant engineer, a specialist (e.g. according to VDI/VDE 2187) or any fully authorized person. This person has access to the complete set of functions of the Frame Application and can use DTM functions without any restrictions. Only OEM-specific operations are locked
Inherits:	Maintenance
User level:	FDTUserInformation.userLevel = planningEngineer
Access verification:	The access as "Planning Engineer" actor requires a password
Use cases:	DTM Instance Handling Simulation [a], Online Operation [a], Repair [a], Offline Operation [a], Bulk Data Handling [a], User Login [a], Online View [a], Audit Trail [a], Archive [a], Report Generation [a], Asset Management [a]

Actor name:	Administrator (abstract actor)
Brief description:	The "Administrator" actor stands for a person that has to perform administrative operations within an engineering environment. The person is responsible for adding and removing of software components
Inherits:	Depends on the Frame Application
User flag [b]:	FDTUserInformation.administrator = TRUE
Access verification:	The access as "Administrator" actor requires a password
Use cases:	Integration and all inherited use cases
Actor name:	**OEM Service (abstract actor)**
Brief description:	The actor "OEM Service" may perform the complete set of DTM-specific functions. OEM-specific operation may be accessible to an "OEM Service" actor, such as resetting of internal counters and exchanging firmware. The "OEM Service" has only inherited use cases, but the inherited use cases (especially the "Online Operation" use case) may have significant extensions
Inherits:	Depends on the Frame Application
User flag [b]:	FDTUserInformation.oemService = TRUE
Access verification:	The access verification for an "OEM Service" shall have two security levels: 1. Frame Application password: enables access to the Frame Application functionality. 2. Device-specific password: enables access to OEM operation with the device. The verification of the device-specific password lies in the responsibility of the DTM or even the device itself
Use cases:	Inherited use cases only
[a] Inherited use cases. [b] "User Flags" may be combined with any "User Level", to specify the inherited actor role of an "Administrator" or "OEM Service" actor.	

5.3 Use cases

5.3.1 Use case overview

Subclause 5.3 describes all use cases of the use case diagram in tabular form. To reduce information, all the attributes of the included use cases, which are identical to the related main use case, are not displayed.

5.3.2 Observation

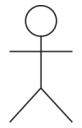


Figure 26 – Observation use cases

Only the observation use cases can be executed by the actor "Observer" (see Figure 26). The "Observer" stands for a person that has the lowest access level. No use case is associated with this person.

5.3.3 Operation

The 'Operation' use cases are available for the actor "Operator" (see Figure 27).

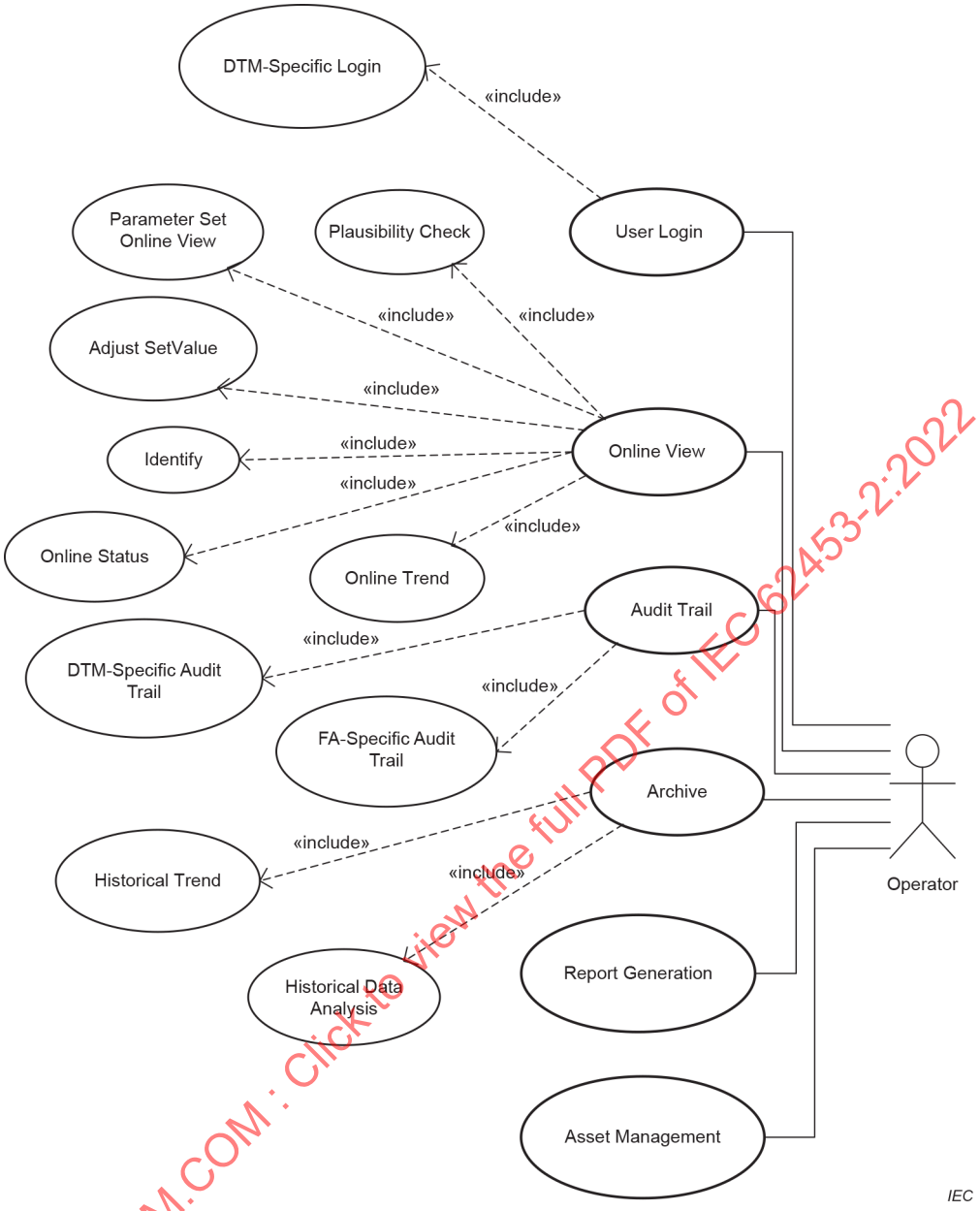


Figure 27 – Operation use cases

Table 7 provides a description of Operation use cases.

Table 7 – Operation use cases

Use case:		**User Login**
Brief description:		A person of specified actor type logs into the Frame Application. If verification fails, the person may act as an "Observer" actor only
GUI:		Part of the Frame Application
Print:		No support
Device connection:		Not established
UserLevel	Observer:	Depends on the Frame Application
	Operator:	Accessible
	Maintenance:	
	Planning Eng.:	
UserFlag	Administrator:	
	OEM Service:	Accessible with extended functionality (see below)
Included use case:		**DTM-Specific Login**
Brief description:		Access to manufacturer-specific information, e.g. production codes, changes log
GUI:		Part of the DTM
Print:		No support
Device connection:		Typically needs an established connection
UserFlag	OEM Service:	Accessible only for OEM Service
Use case:		**Online View**
Brief description:		This use case offers a complete set of operations for viewing of device parameters and status
GUI:		The GUI is part of the DTM. The Frame Application can offer online views for a group of devices
Print:		Online View should always support print function to create documentation
Device connection:		Needs an established connection to one or more devices (Adjust SetValue may influence device data)
UserLevel	Observer:	No access
	Operator:	Accessible
	Maintenance:	
	Planning Eng.:	
UserFlags:		No changes
Included use case:		**Parameter Set Online View**
Brief description:		Enables the actor to load parameters from the device for viewing and documenting (printing). (DTM applicationId: fdtObserve)
Included use case:		**Adjust SetValue**
Brief description:		Enables the actor to adjust the SetValues of a device (e.g. positioner, controllers). (DTM applicationId: fdtAdjustSetValue)
Included use case:		**Online Status**
Brief description:		Use case for viewing and analyzing the current device status. (DTM applicationId: fdtDiagnosis)
Included use case:		**Online Trend**
Brief description:		Use case for generating and watching trends of dynamic online values
Included use case:		**Plausibility Check**
Brief description:		Every time the device parameters or the configuration data is changed, a plausibility check is automatically performed. As long as the plausibility check fails, the data cannot be written to the device
Included use case:		**Identify**
Brief description:		Displays identification of device instance information, e.g. address, TAG, Fieldbus-Rev., DD-Rev., Firmware. This information is protocol dependent. It also may contain device type-specific information. Further information may be provided: references to documentation, area to add comments to the device instance. Refer to Device Identification clause in protocol-specific FDT specifications. (DTM applicationId: fdtIdentify)

Use case:		Audit Trail
Brief description:		Use case to enable the actor viewing and deleting audit trail data
GUI:		The component, which supports audit trail, has to provide an adequate GUI
Print:		Printing for documentation purpose as a need for audit trails and therefore has to be supported
Device connection:		No connection necessary
UserLevel	Observer:	No access
	Operator:	Accessible for viewing only
	Maintenance:	
	Planning Eng.:	View, export and delete
UserFlags:		No changes
Included use case:		**DTM-Specific Audit Trail**
Brief description:		Every DTM might support an audit trail on its own. (DTM applicationId: fdtAuditTrail)
Included use case:		**FA-Specific Audit Trail**
Brief description:		The Frame Application may implement a system global audit trail using internal data and named DTM properties exported from the DTM via the audit trail services
Use case:		**Archive**
Brief description:		The "Archive" use case describes the access to a Frame Application archive for viewing and data analysis
GUI:		The component, which supports archive functions, has to provide an adequate GUI
Print:		Every archive component should support printing too
Device connection:		Not necessary
UserLevel	Observer:	No access
	Operator:	Accessible for viewing only
	Maintenance:	
	Planning Eng.:	View, export and delete
UserFlags:		No changes
Included use case:		**Historical Trend**
Brief description:		The archive operations may support visualization of historical data (for example trending)
Included use case:		**Historical Data Analysis**
Brief description:		Some Frame Applications and DTMs may support extended operations to analyze historical data
Use case:		**Report Generation**
Brief description:		The print function of a use case can normally be started from its GUI. In addition to printing the actual view of a GUI some Frame Applications may implement a configurable documentation mechanism which enables the actor to generate reports. This report may be generated by combining the prints of several use cases or several devices. For example a report could be generated containing "Online View", "Audit Trail" and "Online Operation" printouts for one device or a report may be generated containing the configuration of a communication infrastructure element and its clients
GUI:		Frame Application
Print:		Frame Application in combination with one or more DTMs
Device Connection:		Depends on the type of report
UserLevel, UserFlag:		The printed information may depend on user level and user flags
Use Case:		**Asset Management**
Brief Description:		Frame Application provides mechanism for asset management
GUI:		The component, which supports asset management functions, has to provide an adequate GUI. (DTM applicationId: fdtAssetManagement)
Print:		Frame Application supporting this use case should support printing, too
Device Connection:		Not necessary
UserLevel	Observer:	No access
	Operator:	Accessible for viewing only
	Maintenance:	
	Planning Eng.:	View, export and delete
UserFlags:		No changes

5.3.4 Maintenance

The maintenance use cases are available for the actor "Maintenance Engineer" (see Figure 28).

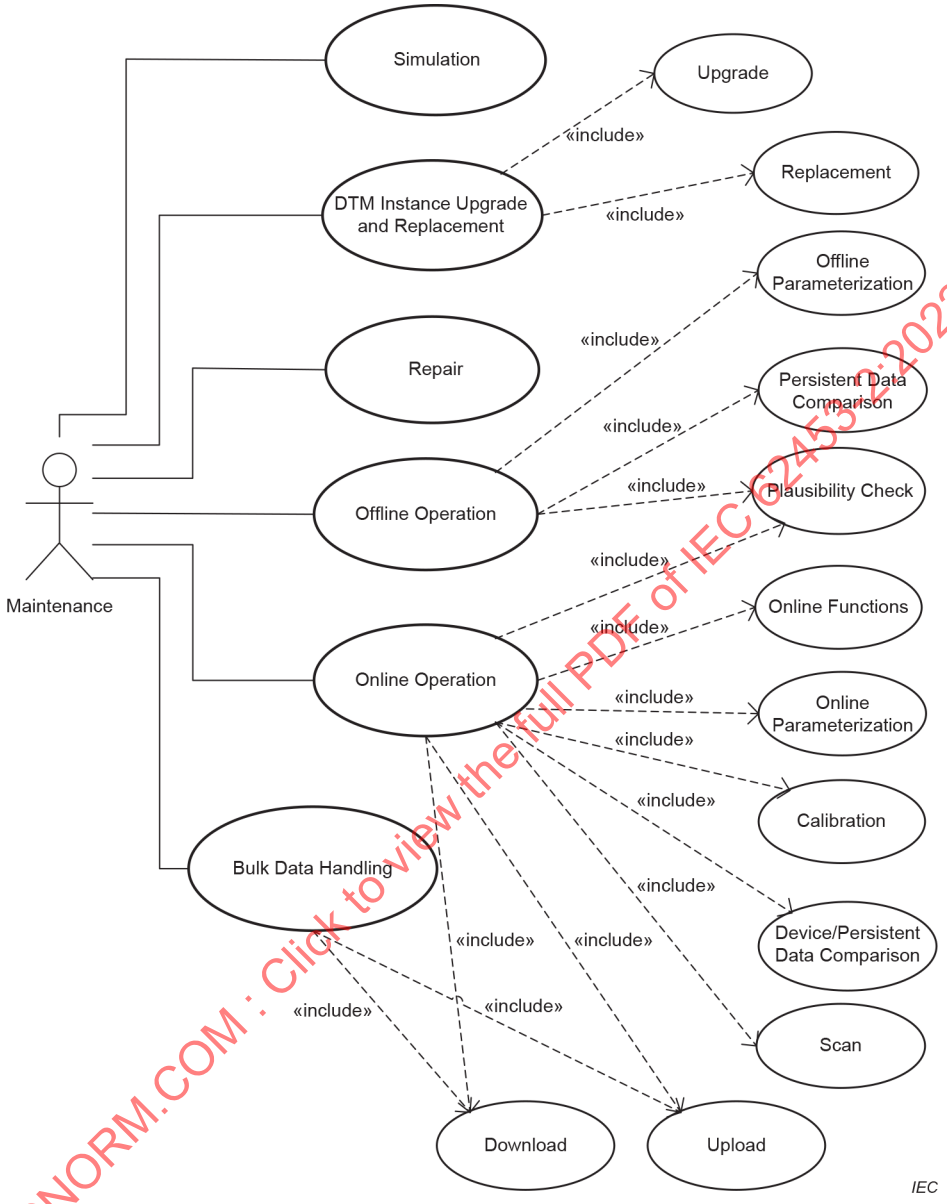


Figure 28 – Maintenance use cases

The Maintenance use cases are described in Table 8.

Table 8 – Maintenance use cases

Use case:		**Simulation**
Brief description:		This use case simulates the behavior of a device. Therefore the actor is allowed to perform temporary changes within the device parameters, like forcing the device to set a fixed current analog output
GUI:		DTM-specific
Print:		DTM-specific
Device connection:		Shall have a connection established
UserLevel	Observer:	No access
	Operator:	
	Maintenance:	Accessible
	Planning Eng.:	
UserFlags:		No changes
Use case:		**Offline Operation**
Brief description:		This use case includes all operations which do not require an established connection to a device. Only for up- and download a connection to a device may be temporarily established
GUI:		DTM and Frame Application
Print:		DTM
Device connection:		Depends on the included use case
UserLevel	Observer:	No access
	Operator:	
	Maintenance:	Accessible
	Planning Eng.:	
UserFlags:		No changes
Included use case:		**Plausibility Check**
Brief description:		Every time the parameter values are changed a plausibility check is automatically performed. As long as the plausibility check fails, the data cannot be saved to the database or written to the device. There may be two options depending on rules or application environment: After display of a warning, inconsistent data may be stored For safety reasons after display of a warning writing of data is prohibited
Users:		The plausibility check may be more tolerant for actors of higher level
Included use case:		**Offline Parameterization**
Brief description:		The user may change parameter values. The changes will only affect data in the Frame Application database after stored as persistent data. Data should be signed as transient data until they are stored
GUI:		DTM (applicationId: fdtOfflineParameterize)
Print:		DTM
Device connection:		No connection necessary
Included use case:		**Persistent Data Comparison**
Brief description:		Allows the user to compare the offline parameter set with parameter sets of other instances, of device default or of Project default parameter sets. The data sets may be editable and data may be transferred from one data set to the other
GUI:		DTM (applicationID: fdtOfflineCompare)
Print:		May be supported by the DTM
Device connection:		Not necessary
Use case:		**Repair**
Brief description:		This use case stands for operations which shall be performed to repair or change a device. Example: A Frame Application supports a temporary removal of a device with automatically downloading parameterization when the device has been reinstalled
UserLevel	Observer:	No access
	Operator:	
	Maintenance:	Accessible
	Planning Eng.:	
UserFlag	Administrator:	No changes
	OEM service:	Accessible with extended functionality (see below)

Use case:		DTM Instance Upgrade And Replacement
Brief description:		This use case stands for operations which shall be performed to upgrade or to replace a DTM. Upgrade or replacement are used when new DTM is different from the previous DTM. The data format of the new DTM can be either compatible (upgrade) or incompatible (replacement) to the data format of the previous DTM
UserLevel	Observer:	No access
	Operator:	
	Maintenance:	Accessible
	Planning Eng.:	
UserFlag	Administrator:	No changes
	OEM Service:	No changes
Included use case:		**Replacement**
Brief description:		This use case stands for operations which shall be performed to replace a DTM in the case of incompatible data set format
Included use case:		**Upgrade**
Brief description:		This use case stands for operations which shall be performed to upgrade a DTM in the case of compatible data set format
Use case:		**Online Operation**
Brief description:		This use case includes all operations which are performed directly with the device
GUI:		DTM-specific
Print:		DTM-specific
Device connection:		Connected or disconnected
Users:	Operator:	No access
UserLevel	Observer:	No access
	Operator:	
	Maintenance:	Accessible
	Planning Eng.:	Fully accessible excluding OEM-specific parameter
UserFlag	Administrator:	No changes
	OEM service:	Accessible with extended functionality (see below)
Included use case:		**Online Functions**
Brief description:		The "online functions" use case offers all procedures which include direct communication with the device. The use case may include simple procedures like resetting but also more complex procedures with several sections like calibration or teach in
Included use case:		**Online Parameterization**
Brief description:		When this use case starts, all parameters are read from the device and exposed to the user within a GUI. Within the GUI, the user may change parameters, depending on the user level. The device is updated immediately if the changed parameterization is in a verified state. (DTM applicationId: fdtOnlineParameterize)
Included use case:		**Device-/Persistent Data Comparison**
Brief description:		This use case gives the user the possibility to compare persistent and device data. The user may edit data and copy between these two sources. (DTM applicationId: fdtOnlineCompare)
Included use case:		**Calibration**
Brief description:		This use case invokes calibration procedures to adjust the input for e.g. pressure transmitters or the current for the output. (DTM applicationId: fdtCalibration)
Included use case:		**Plausibility Check**
Brief description:		Every time the device parameters or the configuration data is changed, a plausibility check is automatically performed. As long as the plausibility check fails, the data cannot be written to the device.
UserLevel, UserFlag:		The plausibility check may be more tolerant for actors of higher level
Included use case:		**Upload**
Brief Description:		Reads the whole parameter set from the device into the Frame Application database. An upload started by an "OEM Service" actor may upload additional OEM parameters
GUI:		If the Frame Application supports up- and download for a group of devices, the Frame Application may offer a GUI for a better control of the upload process
Print:		No support
Device connection:		Needs a temporary connection

Included use case:		Download
Brief description:		Same as for upload, but in the opposite direction
GUI:		If the Frame Application supports up- and download for a group of devices, the Frame Application may offer a GUI for a better control of the download process
Print:		No support
Device connection:		Needs a temporary connection
Included use case:		**Scan**
Brief description:		Retrieves the list of available devices from a Channel object. The Channel object may be provided by a DTM or by Frame Application
GUI:		The Frame Application may offer a GUI for representation of the list of devices
Print:		No support
Device connection:		Needs a temporary connection. (The channel shall have online access)
Use case:		**Bulk Data Handling**
Brief description:		This use case stands for the possibility to handle (e.g. up- and download, parameter adjustment or report generation) a group of instruments at once. This use case handles bulk data on system level
GUI:		Frame Application
Print:		Not supported
Device connection:		Needs a connection
UserLevel	Observer:	No access
	Operator:	
	Maintenance:	Accessible
	Planning Eng.:	
UserFlags:		No changes
Included use case:		**Upload**
Brief description:		Reads the whole parameterization from the devices of the group into the Frame Application database
Included use case:		**Download**
Brief description:		Same as for upload, but in the opposite direction

5.3.5 Planning

The planning use cases are available for the actor "Planning Engineer" (see Figure 29).

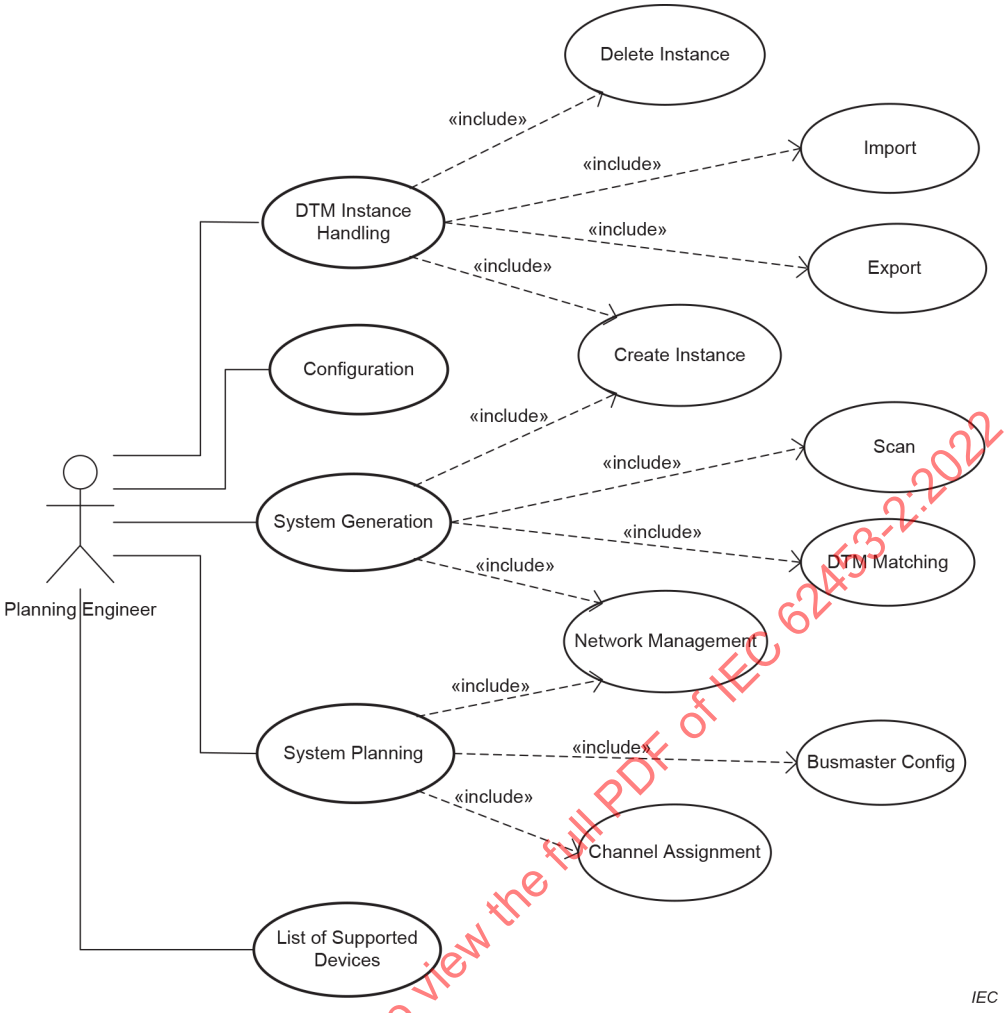


Figure 29 – Planning use cases

Table 9 provides a description of the planning use cases.

Table 9 – Planning use cases

Use case:		**DTM Instance Handling**
Brief description:		With this use case a "Planning Engineer" actor can handle (e.g. instantiate, import, export, delete) a device instance in the Frame Application
GUI:		Frame Application and DTM
Print:		Not supported
Device connection:		Not necessary
UserLevel	Observer:	No access
	Operator:	
	Maintenance:	
	Planning Eng.:	Accessible
UserFlag		No changes
Included use case:		**Create Instance**
Brief description:		In this use case a new device instance can be created and placed into the Project. After creation of a new device instance, the device parameter set shall be instantiated. This action is performed by the DTM
GUI:		Frame Application and DTM (if it supports device default selection)
Included use case:		**Import**
Brief description:		The parameter set may be instantiated by importing data from a database (for example a template database) or by copying the parameter set from an already instantiated and verified device
GUI:		FA

Included use case:		Export
Brief description:		To copy data from one device instance to another, the device instance data can be exported
GUI:		FA
Included use case:		**Delete Instance**
Brief description:		Deletion of a device instance
GUI:		FA
Use case:		**Configuration**
Brief description:		This use case enables the "Planning Engineer" actor to configure a complex device
GUI:		DTM (DTM applicationId: fdtConfiguration)
Print:		May be supported
Device connection:		Shall not have a connection established
UserLevel	Observer:	No access
	Operator:	
	Maintenance:	
	Planning Eng.:	Accessible
UserFlags:		No changes
Use case:		**System Generation**
Brief description:		Use case to perform all necessary actions for the creation of topology based on the result of scan
GUI:		Frame Application and DTM
Print:		Frame Application and DTM
Device connection:		Necessary
UserLevel	Observer:	No access
	Operator:	
	Maintenance:	
	Planning Eng.:	Accessible
UserFlags:		No changes
Included use case:		**Scan**
Brief description:		Retrieves the list of available devices with service scan from a Channel object. The Channel object may be provided by a DTM or by Frame Application
GUI:		The Frame Application may offer a GUI for representation of the list of devices
Print:		No support
Device connection:		Needs a temporary connection. (The channel shall have online access)
Included use case:		**Network Management**
Brief description:		Use case for network management purposes, e.g. address setting as a function of a Communication-DTM. (DTM applicationId: fdtNetworkManagement)
Included use case:		**DTM Matching**
Brief description:		Use case for finding a DTM that matches best the information retrieved in the Scan use case
Included use case:		**Create Instance**
Brief description:		In this use case a new device instance can be created and placed into the Project. After creation of a new device instance, the device parameter set shall be instantiated. This action is performed by the DTM
GUI:		Frame Application and DTM (if it supports device default selection)
Use case:		**System Planning**
Brief description:		Use case to perform all necessary actions for the editing of topology information
GUI:		Frame Application and DTM
Print:		Frame Application and DTM
Device connection:		Not necessary
UserLevel	Observer:	No access
	Operator:	
	Maintenance:	
	Planning Eng.:	Accessible
UserFlags:		No changes
Included use case:		**Network Management**
Brief description:		Use case for network management purposes, e.g. address setting as a function of a communication-DTM. (DTM applicationId: fdtNetworkManagement)
Included use case:		**Bus Master Configuration**
Brief description:		Use case for configuration of bus master DTMs

Included use case:		Channel Assignment
Brief description:		Use case for editing the FDT channel assignments
Use case:		**List of Supported Devices**
Brief description:		In this use case a list of all supported devices within a Project can be viewed and edited. A "Planning Engineer" actor can select a device from this list to create a new device instance. An actor with the "Administrator" actor flag set has access to change this list
GUI:		FA
Print:		No support
Device connection:		No connection
UserLevel	Observer:	No access
	Operator:	
	Maintenance:	
	Planning Eng.:	Accessible for viewing only
UserFlag	Administrator:	Accessible for editing
	OEM Service:	No changes

5.3.6 OEM service

Figure 30 – OEM Service

The "OEM Service" uses cases may provide the "OEM Service" actor (see Figure 30) extended access to use cases of other actor roles.

5.3.7 Administration

The administration use cases are available to users that have additional Administrator permissions (see Figure 31).

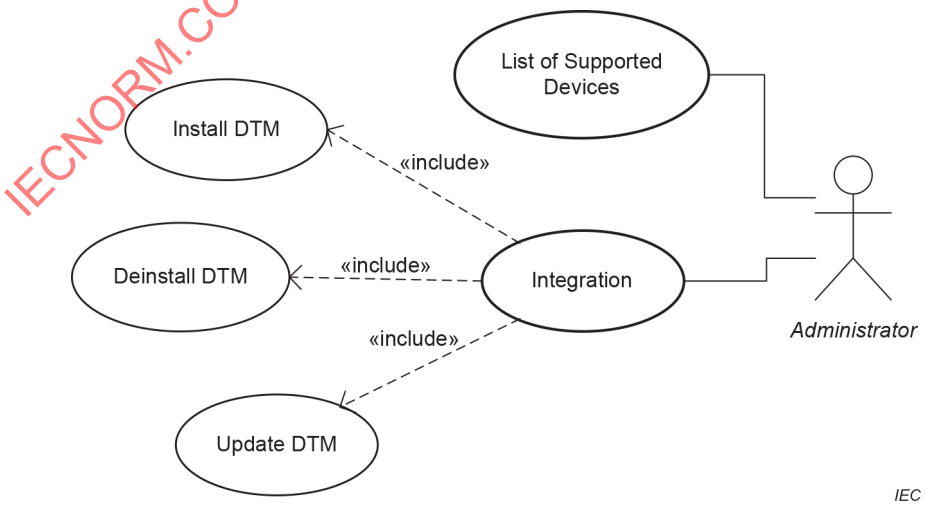


Figure 31 – Administrator use cases

Table 10 provides a description for the Administrator use cases.

Table 10 – Administrator use cases

Use case:		Integration
Brief description:		This use case enables the actor to install, deinstall or update new DTMs. Installation and deinstallation are not controlled by the Frame Application
GUI:		The Frame Application may support the management of DTMs
Print:		Not supported
Device connection:		Shall not have a connection established
Users:	Operator:	Not accessible
	Maintenance:	
	Planning Eng.:	
	Administrator:	Accessible
	OEM Service:	Not accessible
Included use case:		**Install**
Brief description:		Installation of a new DTM
GUI:		Responsibility of the DTM
Included use case:		**Deinstall**
Brief description:		Deinstallation of a DTM
GUI:		Responsibility of the DTM
Included use case:		**Update DTM**
Brief description:		Update/modification of a DTM installation
GUI:		Responsibility of the DTM

6 General concepts

6.1 Address management

Fieldbus systems generally use an addressing concept to distinguish between the different connected devices. Therefore, a DTM generally needs the information about the address of its associated device in order to establish an online connection to it (see service Connect, 7.6.1.2).

A DTM for a device type which can be connected to a system which defines an addressing concept has to provide the NetworkManagementInfo services (see 7.2.11). These services enable to read and write the device addressing information to the DTM. The addressing schema is fieldbus-specific, therefore concrete information which shall be passed to the service is defined by the IEC 62453-3xy documents describing the protocol profile integration in FDT.

It is always the component providing the Communication Channel (DTM or Frame Application) that is responsible for address setting in the linked DTMs. A DTM providing Communication Channels shall support a Presentation object that enables the user to set the addresses (ApplicationID = fdtNetworkManagement, see Table A.4) as shown in Figure 32.

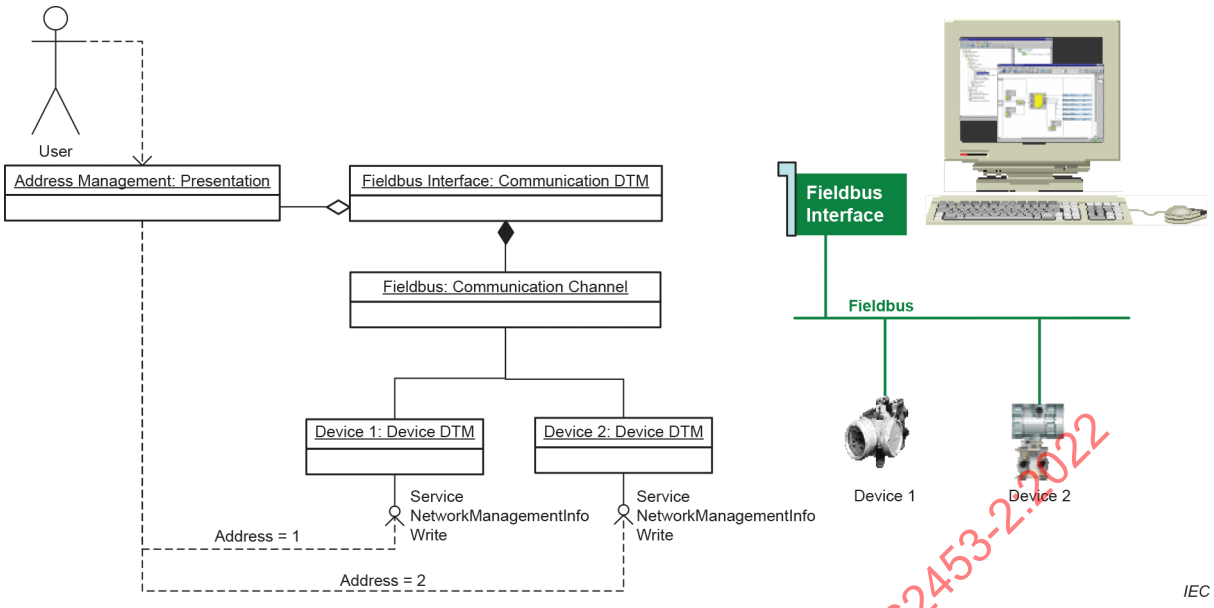


Figure 32 – Address setting via DTM Presentation object

Furthermore, the Communication Channel itself shall support the service SetChildrenAdresses (see 7.6.2.5) to enable the Frame Application to initiate the address setting. The sequence diagrams in 8.2 describe different addressing scenarios in more detail.

How address management is realized if the Communication Channel is provided by the Frame Application is not within the scope of this document.

6.2 Scanning and DTM assignment

6.2.1 Scanning overview

Many fieldbus protocols (or point-to-point communication) define scanning services supporting to request a live list that contains information about connected devices. The information and services described in 6.2 enable the Frame Application to generate or validate corresponding FDT topology without the knowledge of the protocol-specific definitions.

6.2.2 Scanning

Scanning is supported by the Communication Channel service scan (see 7.6.4).

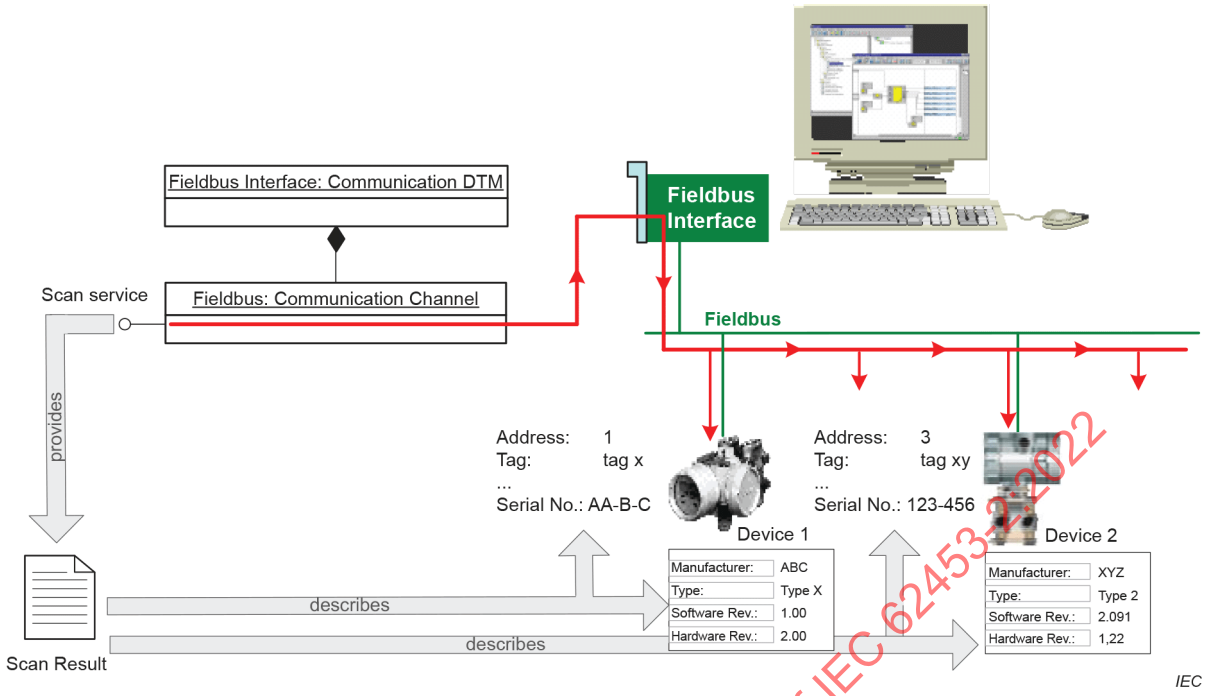


Figure 33 – Fieldbus scanning

The service returns device type and instance related information for all connected devices which can be determined by the means defined by the fieldbus protocol (see Figure 33), for example the device manufacturer, type, hardware and embedded software version or fieldbus address, tag and serial number. This information is fieldbus-specific. The concrete format and transformation to the protocol-independent format which can be used by the Frame Application without fieldbus-specific knowledge is defined by the IEC 62453-3xy document describing the protocol profile integration in FDT.

6.2.3 DTM assignment

A Frame Application may use the information returned by a scan (see 7.6.4) to find proper DTMs that can be added to the FDT topology by comparing the scan information with the device type identification information returned by the service GetIdentificationInformation (see 7.2.3.2) of known DTMs.

Furthermore, a Frame Application may also use the scan result information to validate whether DTMs in an existing FDT topology are the proper ones. This validation is executed by comparing the scan result information with the device type identification information returned by those DTMs. Each element from the scan information can be compared with the DTM device identification information. In this comparison a Frame Application may also apply specific rules.

6.2.4 Manufacturer-specific device identification

Sometimes devices cannot be uniquely identified by the means defined by the fieldbus protocol, for example because the same type ID is used for several different device types of one manufacturer. In this case, the Frame Application may find several DTM Device Types or even DTMs that appear to be the proper ones for devices found during a scan. However, such a device may be identifiable by means defined by the device manufacturer.

FDT enables the manufacturer to implement this specific device identification within a DTM and to allow a Frame Application to use it in a standard way, independently of manufacturer and protocol.

The DTM for such a device shall add manufacturer-specific identification properties to the device identification information that is returned by GetIdentificationInformation (see 7.2.3.2). If the Frame Application tries to find a proper DTM, then it may find only device identifications which match, but have additional elements. The additional elements are not included in the scan result, because they are known only by the device manufacturer. In such a situation, the Frame Application shall look for a DTM Device Type for which the DTMSupportLevel is set to 'identSupport' (see Table A.10 data type idDTMSupportLevel). The 'identSupport'-DTM is added temporarily to the FDT topology in order to request additional information from the device by the service HardwareInformation (see 7.2.3.3). The DTM then shall execute the manufacturer-specific device identification and return the additional device identification elements as a result of the service HardwareInformation. This enables the Frame Application to execute a comparison, which covers all device identification information, have a full match and to replace the 'identSupport'-DTM with the correct DTM and the matching DTM Device Type.

6.2.5 Scan for communication hardware

FDT also defines the means to scan for communication hardware acting as the root element in the FDT topology to access the top-level fieldbus in the system topology or a point-to-point connection.

This mechanism is based on a trial and error principle. The Frame Application just instantiates every registered or pre-selected Communication DTM, invokes HardwareInformation (see 7.2.3.3) and checks whether the service returns information about the found communication hardware or an error.

6.3 Configuration of Fieldbus Master or Communication Scheduler

Many fieldbuses use dedicated devices to control the communication between the different participants, for example the Master-Slave or the Token-Passing concept.

The devices controlling the communication over the fieldbus are called Fieldbus Master or Communication Scheduler. These devices usually need to be configured with the bus parameter information of the devices connected to the fieldbus. In general, the configuration is done with a device-specific configuration tool. The FDT component representing the Fieldbus Master or Communication Scheduler shall provide this configuration tool. That means the configuration tool is either part of the Frame Application or as in Figure 34 part of a DTM.

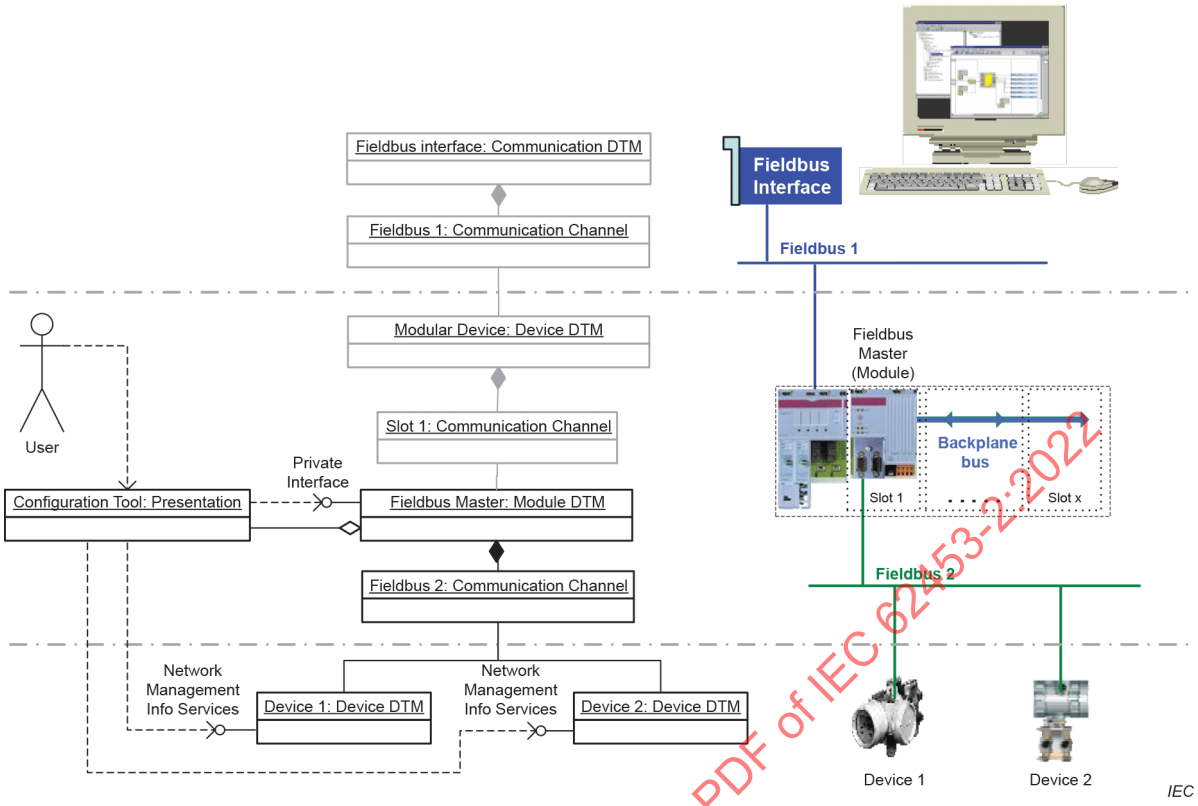


Figure 34 – Fieldbus master configuration tool as part of a DTM

The configuration tool is able to read and write the bus parameter information of the fieldbus participants via the NetworkManagementInfo services (see 7.2.11) of corresponding DTMs. The master or communication scheduler bus parameter information may be accessed by private means defined between the configuration tool and corresponding DTM.

After the configuration tool has set all bus parameter information, each DTM is responsible for writing the information to its device via standard FDT communication operations, for example if a download is requested via service WriteDataToDevice (see 7.2.12.3).

The specific bus parameter information which can be read or written via NetworkManagementInfo services is defined by the IEC 62453-3xy documents describing the protocol profile integration in FDT. Furthermore, the general bus configuration is also described in more detail in the IEC 62453-3xy document, which describes the protocol.

6.4 PLC tool support

6.4.1 General

In a typical PLC system, a fieldbus master hardware periodically communicates the I/O signals with the connected devices and provides them to the PLC in a shared memory area. This memory area is named process image (see Figure 35, simplified by showing only input data). Individual I/O signals are addressed by the PLC application with an offset into this image, while the overall layout of the image is controlled by the fieldbus master configuration software which can be a DTM.

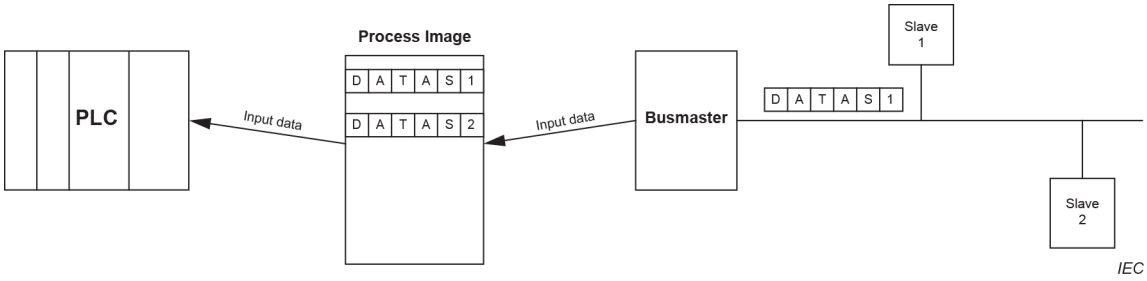


Figure 35 – Process Image

A DTM which represents the fieldbus master shall provide the services to configure the process image. These services enable a PLC Tool Frame Application to request the process image information from the Fieldbus Master DTM (see Figure 36). This information is used for the mapping of variables within a PLC program to the I/O signals in the process image.

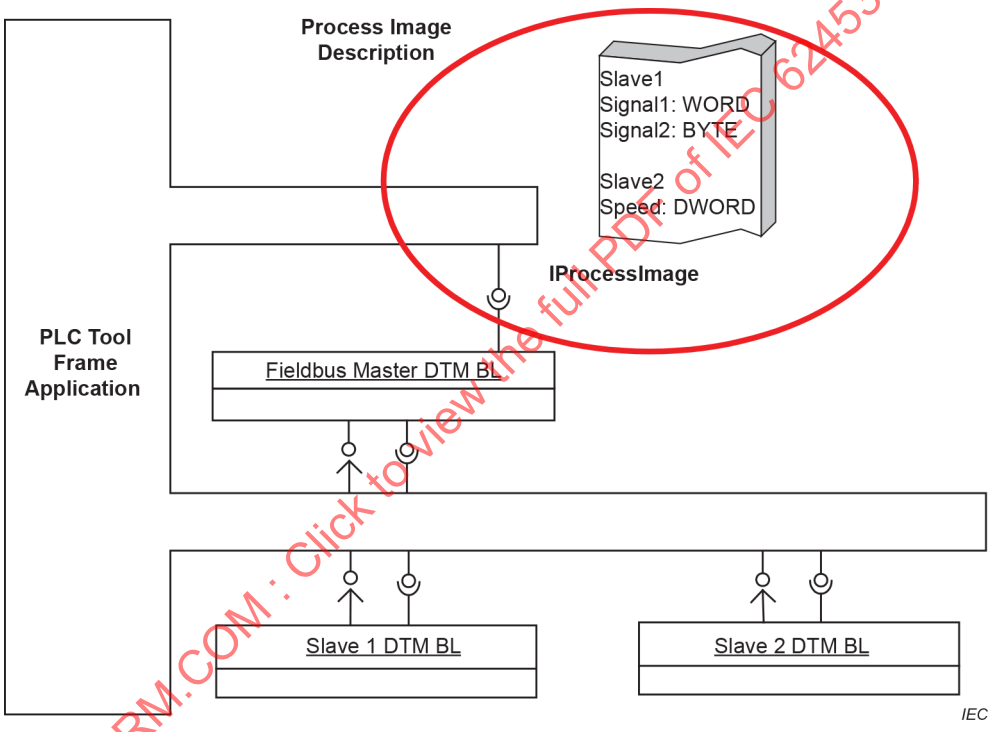


Figure 36 – Transfer of layout information using ProcessImage services

The basic process image information representation is fieldbus-neutral. This allows PLC Tool Frame Applications to do the variable assignment independently of the underlying fieldbus system. In addition, fieldbus-specific I/O signal information (see 6.4.2) is also included in the process image information. This information is gathered by the fieldbus master DTM from the Child DTMs representing the connected devices using the ProcessData services of the Child DTMs (see 4.2.4.2).

6.4.2 Process image modifications while PLC is running

In some automation systems it is a requirement to apply changes to the layout of the process image while the PLC is running (executing control), because the plant process cannot be stopped.

PLC Tool Frame Applications which support such modifications at runtime need to assess if a change can be applied without stopping the PLC. That means, if a user makes a configuration change which would lead to a change in the Child DTM Network Information (see 7.2.11) then this change should be verified before it is applied to the respective field device. This enables the PLC Tool Frame Application to reject changes which would require stopping the PLC if that is not possible at that time.

The validation if the PLC needs to be stopped can only be done in the Fieldbus Master DTM and the PLC Tool Frame Application itself. Thus the Child DTM should support validation of changes in the Network Information by calling the NetworkInfoValidation services of its parent. The Fieldbus Master DTM can then redetermine the resulting process image and validate if it can be applied using the ProcessImageValidation services implemented by a PLC Tool Frame Application (see 7.7.7).

The validation needs to be done before the changes are applied. Only if validation is successful, the Child DTM is allowed to apply the changes.

6.5 Slave redundancy

6.5.1 Redundancy overview

The duplication of critical components of a system with the intention of increasing reliability of the system, usually in the case of a backup or fail-safe, is called redundancy. Figure 37 shows the most common redundancy scenarios in the automation industry.

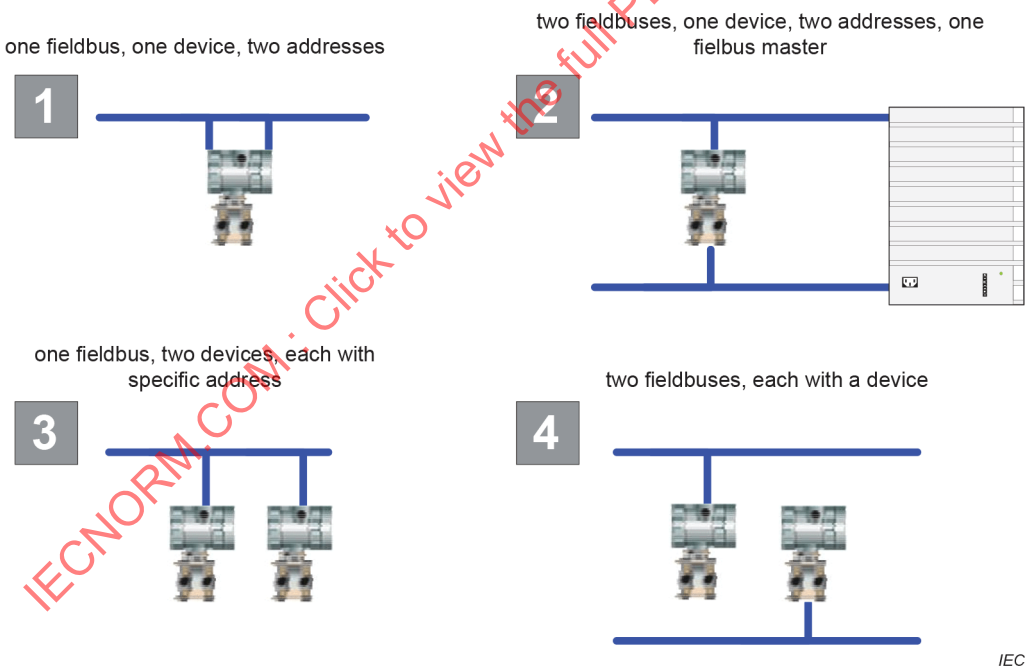


Figure 37 – Redundancy scenarios

The scope of slave redundancy within FDT is one DTM for the configuration of one device connected according to either one of following scenarios:

- scenario 1: one fieldbus using two different addresses;
- scenario 2: connected to two different fieldbuses controlled by one bus master.

Separate redundant devices (scenarios 3 and 4) cannot be handled within FDT, these scenarios would require additional Frame Application functionality, for example with regard to data synchronization.

In scenarios 1 and 2, redundant fieldbuses shall be managed by one redundancy-aware parent component (e.g. a Communication DTM) specific for the appropriate redundant field bus technology. This parent component should typically also be able to handle DTMs for redundant and non-redundant field devices in parallel.

6.5.2 Redundancy support in Frame Application

The DTMs able to handle slave redundancy can be used by any FDT Frame Application without need for specific redundancy support or knowledge. An FDT Frame Application may optionally support the services OnAddedRedundantChild (see 7.7.4.1) and OnRemovedRedundantChild (see 7.7.4.2) to get notified and be able to display redundant slaves in its topology view, for example a field device connected to two different lines or by using specific device icons within a topology view.

Within a Frame Application which is not aware of redundancy, a DTM representing a redundant slave cannot be distinguished within the topology view. But the DTM and the parent component themselves typically display additional information, for example additional fieldbus addresses. Nevertheless, within such a Frame Application, these DTMs provide full functionality with regard to fieldbus redundancy.

As a redundant slave is represented by one DTM instance, no additional functionality with regard to dataset synchronization or multiuser is required.

6.5.3 Parent component for redundant fieldbus

Fieldbus communication for redundant slaves is handled by a fieldbus and redundancy technology-specific parent component, for example a specific Communication DTM. All fieldbuses used to connect redundant slaves shall be handled by one instance of such a parent component. In scenario 2 for example, each fieldbus line is represented by an appropriate Communication Channel of the parent component. Therefore, a Frame Application is able to use such a parent component without knowledge about the physical redundant fieldbus structure.

During execution of service ValidateAddChild (see 7.6.2.3) and service ChildAdded (see 7.6.2.2) the parent component is able to detect if a Device DTM to be added is able to handle redundancy by examining the information that is returned by service NetworkManagementInfoRead (see 7.2.11.1) of the instantiated DTM. In this case, a specific address selection dialog within the parent component can be used. This address selection is fieldbus and redundancy specific. It is up to the parent component if redundancy is handled automatically or only after user interaction.

If the Frame Application is aware of redundancy, the parent component has to inform the Frame Application about the redundant DTM via OnRemovedRedundantChild (see 7.7.4.2).

The parent component shall be able to handle all possible communication paths to the redundant device. Within a service NetworkManagementInfoWrite (see 7.2.11.2) call complete redundant address information can be provided to the DTM instance handling this redundant device. This DTM is then able to use this information to display all available communication paths. During a service Connect (see 7.6.1.2) the DTM provides this complete address information to the parent component. The parent component is then able to select the currently active communication path. The communication path can be changed within the parent component without need for interaction with the Device DTM.

A parent component may also be able to handle a DTM representing a non-redundant field device. In this case, this DTM is handled without using the redundancy-specific data definitions in FDT.

6.5.4 Redundancy support in Device DTM

A DTM able to handle a redundant device provides additional information in service NetworkManagementInfoRead (see 7.2.11.1).

After a service ChildAdded (see 7.6.2.2) call, complete redundant address information can be given by the parent component to the Device DTM. The Device DTM is then able to detect if its appropriate slave is used as a redundant slave. This complete address information shall be used by the Device DTM within a service Connect (see 7.6.1.2), but can also be used for diagnosis and status information. Typically additional redundancy handling is required within the Device DTM itself.

A DTM handling a redundant device shall check all addresses provided by a NetworkManagementInfoWrite (see 7.2.11.2) call. If the Device DTM is not able to handle these addresses, for example because only a vendor-specific offset can be used, an error message should be created and a negative response should be returned to the calling parent component. In this case, the parent component is able to detect Device DTMs which cannot be used at the redundant fieldbus. A Device DTM shall save all redundancy information in its instance data.

A DTM representing a redundant slave can be used as a child of a non-redundant-aware parent component. In this case, the Device DTM shall not use the additional FDT redundancy functionality, GUIs or redundant specific data definitions in FDT.

6.5.5 Scan and redundant slaves

The fieldbus scan provides for each address of a redundant slave one entry. This means a redundant slave cannot be detected.

In scenario 1, mapping of redundant addresses to one instance is only possible for a DTM itself. In scenario 2, mapping can only be done based on project knowledge. Typically, a scan is not executed on a redundant system.

7 FDT service specification

7.1 Service specification overview

Clause 7 specifies the services defined for FDT objects. The service definitions are abstract descriptions and do not represent a specification for implementation. The mapping between the abstract descriptions and the actual implementations derived from these services are defined in technology-specific implementation parts.

The services for each object are organized into service sets. Each service set defines a set of related services. The organization in service sets is a logical grouping used in this specification and a different grouping may be used in the implementation.

Each service is documented with

- a service abstract;
- a table for request and response values;
- a service description;
- a state availability statement.

The service abstract gives a brief overview.

Within the table for request and response values, the arguments which will be used for the service are described. For each argument (see Annex A for a description of the argument data types) it is defined whether the argument is mandatory ('M') or optional ('O'), used for the service call (Request) or as return value (Response) or not needed ('-').

The description of the service defines the behaviour and usage of the service.

The state availability statement defines for each service whether the service is available in state 'configured' or 'communication allowed'.

This document does not define whether it is optional or mandatory to implement the defined services. This definition is provided by the technology-specific implementation parts. Each technology-specific part provides an annex describing the actual implementation of the services (for example mapping of services to interface methods).

NOTE The mechanism for informing about not implemented services can be technology specific. For some technology, not implemented service can be recognized by the client if a server does not provide certain interfaces. For other technologies, not implemented services can be recognized by special service responses.

Whether the services are executed in a synchronous or asynchronous model depends on the implementation technology. 'Synchronous' means that the service caller is blocked during the complete execution time of the service. 'Asynchronous' means that the caller is able to continue with other operations until service execution is complete. The used service execution model is defined in the object model integration profile document describing the mapping to certain implementation technology. Both execution models may be used together for the implementation of services. In addition, canceling of an active service execution may be defined.

7.2 DTM services

7.2.1 General services

7.2.1.1 Service PrivateDialogEnabled

This service is used by the FrameApplication to notify a DTM whether it is allowed to open a private dialog window. The arguments for the service are shown in Table 11.

Table 11 – Arguments for service PrivateDialogEnabled

	Request	Response
Enabled (Boolean)	M	–

This special condition is set by a Frame Application during runtime. This may be necessary for instance if

- a running user action shall not be disturbed;
- it is not allowed that a dynamically opened window gets the focus or hides other windows, or
- the DTM is executed on a different host than the GUI.

Enabled set to FALSE means that the DTM shall not open any GUI other than GUIs that are opened by the FA services (e.g. service OpenPresentationRequest and service UserDialog). If under this condition any error occurs, the DTM is still able to send a notification to the Frame Application via service OnErrorMessage (see 7.7.2.1).

If the DTM needs a user action, and therefore has to open a user dialog, it should call the Frame Application service UserDialog (see 7.7.8.3). If no GUI can be shown, the default behavior at this request is not to open the dialog and to provide the DTM with a default return value. So it is up to the Frame Application to allow opening the dialog, delay the request until the current user action is finished, or to refuse the request by sending the default response.

DTMs should inform the user via service UserDialog if a specific functionality cannot be performed because private dialogs are not allowed.

Examples for private dialogs are:

- message boxes (i.e. standard message box);
- file or printer selection dialogs (i.e. provided by operating system);
- (default) web browsers;
- (default) mail clients;
- help file view;
- manual viewer;
- splash screens;
- external stand-alone applications,
- any other windows.

If the dialogs are opened under control of the Frame Application, they are defined not to be private dialogs.

Examples are:

- dialogs opened by service UserDialog;
- DTM GUI opened by service OpenPresentationRequest;
- applications started by service OpenPresentation;
- windows opened by the Frame Application due to a <Document> entry in the response received from GetFunctions.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.1.2 Service SetLanguage

This service is used by the Frame Application to notify a DTM during initialization about the language to use in all Presentation objects and for user or error messages. The arguments for the service are shown in Table 12.

Table 12 – Arguments for service SetLanguage

	Request	Response
LanguageID	M	-

The Frame Application sets the language during initialization of the DTM, so that all Presentation objects of the same instance have the same language. Also, the messages on the event service such as service OnErrorMessage (see 7.7.2.1) and the human readable contents returned by services GetTypeInformation (see 7.2.3.1) or GetDocumentation (see 7.2.7) shall use the requested language. If a DTM does not support the requested language, it shall use the current language (if it is already initialized) or on the first initialization set the current language to its default language.

The supported languages of a DTM are listed in the information returned by service GetTypeInformation (see 7.2.3.1). One DTM instance is always initialized with one language. It is up to a DTM whether it is able to change the current language of a Presentation object while it is shown. The output format for numbers, currencies, times, and dates shall be based on the regional options of the operation system.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.1.3 Service SetSystemGuiLabel

This service is called by the Frame Application to set a unique human readable identifier of the DTM instance in the context of the Frame Application. The arguments for the service are shown in Table 13.

Table 13 – Arguments for service SetSystemGuiLabel

	Request	Response
windowTitle	M	-

This service is called by the Frame Application in order to set a system label, for example for a message box or a Presentation object which is part of the DTM and is not to be embedded within a Frame Application user interface. The Frame Application shall set the unique human readable identifier of the DTM instance in the context of the Frame Application which guarantees a unique identification between the device and, for example, a message box of a DTM. The DTM shall use this system label for all kinds of windows that will be opened by the DTM themselves. The DTM may extend this title with specific information. The Frame Application has to call this service as early as possible. As long as the service is not called, the DTM has to use the tag of the device as a human readable string by default.

The human readable identifier shall not be stored by the DTM. It is mandatory to update the labels of all open windows when this service is called.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.2 DTM services related to installation

Each DTM which should be visible for integration within the Frame Application shall be registered within the system. In addition to the registration, the DTM shall give information as to whether the DTM represents a device, module or block.

DTMs shall inform the system whenever the DTM is:

- installed;
- uninstalled;
- modified, (e.g. new device types are supported or the DTM was updated (bug fix)).

By using this information, a Frame Application is able to build a database with all available DTMs on the system. The detailed implementation depends on the used technology and is described in an object model integration profile document.

7.2.3 DTM services related to DTM/device information

7.2.3.1 Service GetTypeInformation

Returns general DTM type information like name, version, vendor and supported DTM Device Types. The arguments for the service are shown in Table 14.

Table 14 – Arguments for service GetTypeInformation (for DTM)

	Request	Response
fdt:VersionInformation	–	M
fdt:Version	–	M
fdt:DtmDeviceTypes	–	M

This service provides access to DTM-specific information. This data are available as soon as the DTM is instantiated. The DTM does not need any instance-specific data to provide this information.

Usually, this information is used by the Frame Application to create libraries, to select a DTM, or for simple validations.

If the service is called at a BTM, then the following service request and response parameters are used (see Table 15).

Table 15 – Arguments for service GetTypeInformation (for BTM)

	Request	Response
fdt:VersionInformation	–	M
fdt:Version	–	M
btm:BtmBlockTypes	–	M

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.3.2 Service GetIdentificationInformation

This service requests device identification information for specified DTM Device Type and protocol. The arguments for the service are shown in Table 16.

Table 16 – Arguments for service GetIdentificationInformation (for DTM)

	Request	Response
fdt:DtmDeviceType	M	–
fdt:ProtocolId	M	–
fieldbus:DeviceTypeIdentifications or devIdent:DeviceTypeIdentificationList	–	M

This service returns device identification information for a requested device or block type. The response may contain also protocol-specific information.

If the service is called at a Communication-DTM, then the response contains devIdent:DeviceTypeIdentification List.

If the service is called at a BTM, then the following service request and response parameters are used (see Table 17).

Table 17 – Arguments for service GetIdentificationInformation (for BTM)

	Request	Response
btm:BtmBlockType	M	–
fdt:ProtocolId	M	–
fieldbus: DeviceTypeIdentifications	–	M

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.3.3 Service HardwareInformation

This service requests scan for availability of hardware and the corresponding identification information. The arguments for the service are shown in Table 18.

Table 18 – Arguments for service Hardware information (for DTM)

	Request	Response
ScanIdentificationList	–	M
ScanIdentificationList may be either ident:ScanIdentificationList or fieldbus:ScanIdentifications.		

The Frame Application executes this function to check if corresponding hardware is responsive and to request identification information.

This service is available in states:
[_] 'configured';
[X] 'communication allowed'.

7.2.3.4 Service GetActiveTypeInfo

The DTM shall provide information about the selected DTM Device Type. The arguments for the service are shown in Table 19.

Table 19 – Arguments for service GetActiveTypeInfo

	Request	Response
fdt:DtmDeviceType	–	M
net:DtmDeviceInstanceTopology	–	O

The service shall always return the current DTM Device Type. That means in case of an update of the DTM, the changed DTM Device Type shall be returned (e.g. higher software version) instead of the DTM Device Type given during the service Initialize (see 7.2.4.1) call.

Optionally, the service returns information about the internal structure of the corresponding device. The returned information is protocol- and device-specific.

If the service is called at a BTM, then the following service request and response parameters are used (see Table 20).

Table 20 – Arguments for service GetActiveTypeInfo (for BTM)

	Request	Response
btm:BtmBlockType		M -

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.4 DTM services related to the DTM state machine

7.2.4.1 Service Initialize

The service initializes the DTM in order to bring it to a working state. The arguments for the service are shown in Table 21.

Table 21 – Arguments for service Initialize (for DTM)

	Request	Response
fdt:DtmDeviceType	M	–
user:FDTUserInformation	M	–
fdt:FrameVersion	O	–
fdt:SystemTag	M	–
fdt:FrameObjectReference	M	–
fdt:serviceSucceeded	–	M

The DTM receives information regarding the device type that it will represent, regarding the user that is going to access the DTM, regarding the runtime environment, the identifier of the DTM and a reference to the Frame Application itself.

In general, it is expected that a DTM adapts the provided functionality according to the role of the current user.

Also, it is expected that the DTM adapts its behaviour regarding the FDT functionality of the runtime environment (according to FrameVersion).

If the service is called at a BTM, then the following service request and response parameters are used (see Table 22).

Table 22 – Arguments for service Initialize (for BTM)

	Request	Response
btm:BtmBlockType	M	–
user:FDTUserInformation	M	–
fdt:FrameVersion	O	–
fdt:SystemTag	M	–
fdt:FrameObjectReference	M	–
fdt:serviceSucceeded	–	M

This service is available in states:
[X] 'initial state' ;
[_] 'configured';
[_] 'communication allowed'.

Calling this service leads to a state transition to state 'configured'.

7.2.4.2 Service SetLinkedCommunicationChannel

The service informs the DTM about the linked Communication Channel within the FDT topology which shall be used to communicate with its device. The arguments for the service are shown in Table 23.

Table 23 – Arguments for service SetLinkedCommunicationChannel

	Request	Response
fdt:CommunicationObjectReference	M	–
fdt:serviceSucceeded	–	M

The Communication Channel is set by the Frame Application: the DTM is able to check the supported communication protocols by calling the service GetSupportedProtocols (see 7.6.1.1) and the GatewayBusCategory information by calling the ReadChannelData (see 7.5.1). In general, the Frame Application is responsible for establishing a valid link between a Communication Channel and a DTM (see 4.3), but this check may be for example necessary if a DTM supports multiple protocols and wants to find out which shall be used.

This service is available in states:
[X] 'configured';
[_] 'communication allowed'.

7.2.4.3 Service EnableCommunication

The service allows or permits a DTM to communicate with its device. The arguments for the service are shown in Table 24.

Table 24 – Arguments for service EnableCommunication

	Request	Response
Boolean	M	–
fdt:serviceSucceeded	–	M

The service is called by the Frame Application with TRUE to bring the DTM from the state 'configured' into the state 'communication allowed'. Afterwards, the DTM is allowed to establish a communication link to its device by calling the Communication Channel service Connect (see 7.6.1.2).

The service is called with FALSE to bring the DTM from the state 'communication allowed' back into the state 'configured'. If the DTM has accepted the call (fdtServiceSucceeded = TRUE), then all established communication links shall be closed (see services Disconnect (7.6.1.3) or AbortRequest (7.6.1.4)). The DTM is allowed to reject the (fdt:serviceSucceeded = FALSE) if communication service calls are active which cannot be terminated.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.4.4 Service ReleaseLinkedCommunicationChannel

The service informs the DTM that it shall release the Communication Channel set by the service SetLinkedCommunicationChannel (see 7.2.4.2). The arguments for the service are shown in Table 25.

Table 25 – Arguments for service ReleaseLinkedCommunicationChannel

	Request	Response
fdt:serviceSucceeded	–	M

This service is available in states:
[X] 'configured';
[_] 'communication allowed'.

7.2.4.5 Service ClearInstanceData

Used to inform the DTM that it has to clean up, for example its log files or protocols. The instance data will be deleted by the Frame Application. The arguments for the service are shown in Table 26.

Table 26 – Arguments for service ClearInstanceData

	Request	Response
fdt:serviceSucceeded	–	M

The service is used to inform a DTM that it has to clean up, for example its log files or protocols in its private storage (see 4.9). After this service call, the instance data will be deleted by the Frame Application. The Frame Application is responsible for ensuring the pre-conditions for the delete. That means the Frame Application shall ensure that all started DTM instances related to this instance data are terminated.

This service is available in states:
[X] 'configured';
[_] 'communication allowed'.

7.2.4.6 Service Terminate

This service informs the DTM that it has to release its references to other components. The DTM will be terminated by the Frame Application. The arguments for the service are shown in Table 27.

Table 27 – Arguments for service Terminate

	Request	Response
fdt:serviceSucceeded	–	M

The DTM has to release all references to other components and has to terminate all pending or running functions. It shall also close all user interfaces. It is up to a DTM to decide if transient data should be stored or not.

This service is available in states:
[X] 'configured';
[_] 'communication allowed'.

7.2.5 DTM services related to functions

7.2.5.1 Service GetFunctions

This service returns information about standard (defined by ApplicationID) or additional functionalities (defined by FunctionId) supported by a DTM. The arguments for the service are shown in Table 28.

Table 28 – Arguments for service GetFunctions

	Request	Response
fdt:OperationPhase	M	–
func:Functions	–	M

This service provides information about DTM functions that enable the Frame Application for example to create a DTM-specific menu. This data is available as soon as the DTM is instantiated but the information may change if it is instance specific.

A DTM shall inform the Frame Application about any function changes (e.g. number, status, label, etc.) by calling the service OnFunctionChanged (see 7.7.2.4).

Information about how to execute functions with user interface shall be requested by service GetGuiInformation (see 7.2.5.3) and function calls without user interface shall be started by service InvokeFunctions (see 7.2.5.2).

The service additionally provides access to documents provided by a DTM, which can be displayed by the Frame Application or a special viewer program.

It is expected that a DTM adapts the provided list of functions according to the role of the current user. As long as the DTM does not have valid user information, it should behave as if the user with the least rights ("Observer") is using the DTM.

It is possible that functions with associated module Ids (so-called module functions) will not be shown directly in the standard context menu of the DTM node. They may appear in a submenu which is associated to a DTM module or in the context menu of a DTM module node. In order to ensure that module functions can be called by the user, the Frame Application should provide a submenu "Modules" in the context menu of the DTM node or provide appropriate context menus for DTM modules or offer these functions by other means.

The DTM has to ensure that all accessible module functions (enabled and not hidden) are valid. That means the Frame Application is not responsible for matching the given module Ids to the list of the existing modules returned by service GetActiveTypeInfo (see 7.2.3.4).

If the DTM disables a group of functions (function data type), it shall also disable all functions (function data type) below that group if this is the intended behavior. If the DTM does not disable sub-functions, the Frame Application can still make use of them.

If a Frame Application does not support the operation phases ('notSupported'), the DTM should provide all functions based on the current state and the current user role.

If the DTM wants to limit the number of opened user interfaces, it should disable the corresponding functions. If a user interface is closed, the DTM has to enable the function again.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.5.2 Service InvokeFunctions

This service starts execution of a function of a DTM without user interface. The arguments for the service are shown in Table 29.

Table 29 – Arguments for service InvokeFunctions

	Request	Response
func:FDTFunctionCall List	M	–

This service executes a DTM function provided without a user interface. Information about functions supported by a DTM is returned by service GetFunctions (see 7.2.5.1).

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.5.3 Service GetGuiInformation

This service provides information on how to start the user interface of a DTM. The arguments for the service are shown in Table 30.

Table 30 – Arguments for service GetGuiInformation

	Request	Response
func:FDTFunctionCall	M	–
func:GUIInformation	–	M

It returns information that enables the Frame Application to instantiate the user interface. Depending on the returned information, the Presentation object may be a type of object that allows integration into the GUI of the Frame Application (see 7.7.8.1) or it may be an object that runs outside of the GUI of the Frame Application (see 7.2.5.4).

Integration and interaction between these objects is technology dependent and defined within object model integration profile documents of the IEC 62453 series.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.5.4 Service OpenPresentation

This service requests the DTM to open a user interface for a specific function call. The arguments for the service are shown in Table 31.

Table 31 – Arguments for service OpenPresentation

	Request	Response
fdt:invokeID	M	
func:FDTFunctionCall	M	
fdt:windowTitle	M	
fdt:DisplayContext	O	
fdt:serviceSucceeded	–	M

The FDTFunctionCall requests the opening of the corresponding Presentation object. In general, it is up to the Frame Application to determine the passed FDTFunctionCall and the DTM decides the kind of presentation.

This service shall always bring a user interface to the foreground, or at least an error message. Already started Presentation objects, identified by the InvokeID, shall be popped to the foreground. The request of an already started Presentation object with a new InvokeID shall be rejected by the DTM.

The InvokeID is used by a Frame Application for the association within the service ClosePresentation (see 7.2.5.5) request. Furthermore, it allows the Frame Application to handle a list of open user interfaces.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.5.5 Service ClosePresentation

Request to a DTM to close a user interface identified by InvokeID. The arguments for the service are shown in Table 32.

Table 32 – Arguments for service ClosePresentation

	Request	Response
fdt:invokeID	M	
fdt:serviceSucceeded	–	M

The DTM just checks whether it is able to close the user interface or not. If it is able to close the user interface, the service returns success and the DTM shall start its shut down procedure for the user interface. During this shutdown, it has to unlock its instance data and release the online connection to its device if necessary. The DTM itself is not terminated.

In case of errors, the DTM shall supply further details by calling the service OnErrorMessage (see 7.7.2.1).

The InvokeID is used by a Frame Application for the association to the appropriate service OpenPresentation (see 7.2.5.4) request.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.6 DTM services related to Channel objects – service GetChannels

This service returns the Channel objects of a DTM. The arguments for the service are shown in Table 33.

Table 33 – Arguments for service GetChannels

	Request	Response
ChannelObjectReference List	–	M

This service returns the channels of a DTM. For simple devices, a Channel object provides only the information for channel assignment.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.7 DTM services related to documentation – service GetDocumentation

This service provides the DTM-specific documentation for a specific DTM function. The arguments for the service are shown in Table 34.

Table 34 – Arguments for service GetDocumentation

	Request	Response
func:FDTFunctionCall	M	–
doc:Documentation	–	M

This service shall return information which can be used directly for documentation purposes. The content of the returned information is defined by the passed function call Id, which is available via the service GetFunctions (see 7.2.5.1). Only functions with the attribute Printable shall be supported. The Frame Application shall transform the returned information to a user readable format.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.8 DTM services to access the instance data

7.2.8.1 General

These services enable a Frame Application the access to device parameters. The DTM provides its current in-memory representation of its instance data. It is up to a DTM and depends on the device type and fieldbus type which parameters are available.

7.2.8.2 Service InstanceDataInformation

This service returns information about the device-specific parameters. The arguments for the service are shown in Table 35.

Table 35 – Arguments for service InstanceDataInformation

	Request	Response
fdt:DatasetState	–	M
fdt:StorageState	–	M
param: DtmItemInfoList	–	M
fdt:ModifiedInDevice		M

This service provides information about the device-specific parameters and state. The information may contain information related to configuration parameters as well as asset management related data or can be empty for devices without public data. The contents of the provided information may also depend on the current configuration of the device and the user roles.

Several responses may be provided by the DTM if the list or status of parameters changes.

The service provides the transient data of a DTM. That means, if the DTM is active or has for example an open user interface the provided parameter can differ from the actual stored instance data. The state of the received data is classified.

Parameters provided may also be available as Channel objects (provided by service ReadChannelData (see 7.5.1)). The relation of these items can be identified by the information 'SemanticId' (see Table A.4).

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.8.3 Service InstanceDataRead

The service provides read access to DTM instance data. The arguments for the service are shown in Table 36.

Table 36 – Arguments for service InstanceDataRead

	Request	Response
param:ItemId list	M	–
param:DtmItem list	–	M

Via this service a Frame Application is able to request data from the DTM instance data.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.8.4 Service InstanceDataWrite

The service provides write access to DTM instance data. The arguments for the service are shown in Table 37.

Table 37 – Arguments for service InstanceDataWrite

	Request	Response
param:DtmItem list	M	M

Via this service, the Frame Application requests a DTM to write the specified data to its instance data. Furthermore, the DTM has to apply the business rules in order to keep the instance data consistent.

The response contains the device data of the successfully written data specified by the request (it may differ compared to the written value due to, for example rounding procedures within the device).

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.9 DTM services to evaluate the instance data

7.2.9.1 Service Verify

This service validates the complete instance data according to the internal business rules of the DTM. The arguments for the service are shown in Table 38.

Table 38 – Arguments for service Verify

	Request	Response
fdt:serviceSucceeded	–	M

A Frame Application calls this service typically to ensure a consistent dataset, for example after persistent load or before going online.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.9.2 Service CompareDataValueSets

This service compares the instance data of an external DTM supporting the same DatasetId with its own. The arguments for the service are shown in Table 39.

Table 39 – Arguments for service CompareDataValueSets

	Request	Response
fdt:SystemTag	M	–
fdt:serviceSucceeded	–	M

The structure and the parameter values for configuration, parameterization and identification are relevant for the comparison. The runtime-dependent parameters (e.g. operation hours) of the data are not relevant for comparison.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.10 DTM services to access the device data

7.2.10.1 General

These services provide a Frame Application online access to the specific parameters of a device. The DTM shall be prepared for multiple asynchronous requests in parallel. The requests shall be processed in the order received.

The service shall not modify the instance data.

7.2.10.2 Service DeviceDataInformation

This service provides a list of the available device-specific parameters and process values. The arguments for the service are shown in Table 40.

Table 40 – Arguments for service DeviceDataInformation

	Request	Response
param:DtmItemInfo list	–	M

This service provides a list of the available device-specific parameters and process values. Within a DTM this list may contain items related to configuration parameters, process values as well as asset management related data such as stroke counter. In the DTM state 'configured' the returned item list is based on the current instance data, which could be different from the configuration of the device. In this case, services DeviceDataRead (see 7.2.10.3) and DeviceDataWrite (see 7.2.10.4) may fail.

The service provides a list of items that can be read or written from/to the DTM via DeviceDataRead or written to the DTM via DeviceDataWrite. The source for this data is the device itself.

Items provided within this list may also be available from the channel service, service ReadChannelData (see 7.5.1) or DTM service InstanceDataInformation (see 7.2.8.2). The related items can be identified via the 'SemanticId' (see Table A.4).

The information (data items with access permissions) may depend on the current configuration of the device and on the current user role. Several responses may be provided by the DTM if list or status of parameters is changed.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.10.3 Service DeviceDataRead

This service provides read access to device data. The arguments for the service are shown in Table 41.

Table 41 – Arguments for service DeviceDataRead

	Request	Response
param:ItemId list	M	–
param:DtmItem list	–	M

This service enables a Frame Application to request data from a device. Error information shall be handed over to the Frame Application by the related response.

Execution of the service shall not change the data of the instance data.

The DTM shall always accept the request. If the request cannot be processed, the reason for failure shall be provided as part of the response.

The DTM shall be able to handle more than one request at a time.

This service is available in states:
[_] 'configured';
[X] 'communication allowed'.

7.2.10.4 Service DeviceDataWrite

This service provides write access to device data. The arguments for the service are shown in Table 42.

Table 42 – Arguments for service DeviceDataWrite

	Request	Response
param:DtmItem list	M	M

This service enables the Frame Application to request a DTM to write the specified data to its device according to the device-specific rules. Error information shall be handed over to the Frame Application by the related response.

Execution of this service shall not change the data of the instance data.

The DTM has to check whether it could manipulate the flag 'ModifiedInDevice' (see Table A.4) in the instance data by requesting a lock. If the lock request fails, the DTM has also to refuse the DeviceDataWrite call – an appropriate response shall be provided.

The DTM shall always accept the request. If the request cannot be processed, the reason for failure shall be provided asynchronously as part of the response.

The DTM should be able to handle more than one request at a time.

The response as DtmItem list contains the device data of the successfully written data (may differ from the written value due to, for example, rounding procedures within the device).

This service is available in states:
[_] 'configured';
[X] 'communication allowed'.

7.2.11 DTM services related to network management information

7.2.11.1 Service NetworkManagementInfoRead

This service provides read access to network management information. The arguments for the service are shown in Table 43.

Table 43 – Arguments for service NetworkManagementInfoRead

	Request	Response
net:NetworkInfo	–	M

This service provides write access to network management information. The returned information is protocol-specific. It contains for example the device bus address, tag and further bus specific configuration settings.

The usage of this service is described in more details in 6.1.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.11.2 Service NetworkManagementInfoWrite

This service provides write access to network management information. The arguments for the service are shown in Table 44.

Table 44 – Arguments for service NetworkManagementInfoWrite

	Request	Response
net:NetworkInfo	M	–

This service provides write access to network management information which can be read by service NetworkManagementInfoRead (see 7.2.11.1).

The usage of this service is further described in 6.1.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.12 DTM services related to online operation

7.2.12.1 Service DeviceStatus

This service provides information about the status of the device. The arguments for the service are shown in Table 45.

Table 45 – Arguments for service DeviceStatus (for DTM)

	Request	Response
status:DTMDeviceStatus	–	M

The DTM shall load the current status from the device. Depending on the fieldbus protocol, the DTM may additionally upload its current diagnosis information. Depending on this information, the DTM shall provide status in a human readable format. The function shall not open a user interface to allow the check of complete networks.

A BTM shall return the status of the related block.

This service is available in states:
[_] 'configured';
[X] 'communication allowed'.

7.2.12.2 Service CompareDataValueSetWithDeviceData

This service compares DTM instance data with device data. The arguments for the service are shown in Table 46.

Table 46 – Arguments for service CompareInstanceDataWithDeviceData (for DTM)

	Request	Response
onlineComp:DTMOnlineCompare	–	M

This service is used for batch processing and shall work without user interface. It compares its instance data with the device data uploaded from its device. If the DTM is able to upload the data from the device, then it shall compare the data and return whether the data is equal or not. Otherwise, the response shall contain communication error information.

If the DTM has no comparable online data, it shall return 'noComparableData' as value of the 'StatusFlag'.

Comparison should only include data significant for the configuration, parameterization and identification of the device. Data related to runtime (e.g. operation hours) should not be included.

This service is available in states:
[_] 'configured';
[X] 'communication allowed'.

7.2.12.3 Service WriteDataToDevice

This service requests to write online data to the device. The arguments for the service are shown in Table 47.

Table 47 – Arguments for service WriteDataToDevice (for DTM)

	Request	Response
fdt:serviceSucceeded	–	M

This service is initiated by the Frame Application. It is mandatory for all DTMs that support online data. The service is used to write all the parameters needed for commissioning of the device from the DTM's instance data to the device.

In case of errors, the DTM shall inform the Frame Application via the service OnErrorMessage (see 7.7.2.1).

This service is available in states:
[_] 'configured';
[X] 'communication allowed'.

7.2.12.4 Service ReadDataFromDevice

This service requests to read online data from the device. The arguments for the service are shown in Table 48.

Table 48 – Arguments for service ReadDataFromDevice(for DTM)

	Request	Response
fdt:serviceSucceeded	–	M

This service is initiated by the Frame Application. It is mandatory for all DTMs that support online data. The service is used for the DTM to read all the parameters needed for commissioning of the device from the device into the DTM's instance data.

In case of errors, the DTM shall inform the Frame Application via the service OnErrorMessage (see 7.7.2.1).

This service is available in states:
[_] 'configured';
[X] 'communication allowed'.

7.2.13 DTM services related to data synchronization

7.2.13.1 Service OnLockInstanceData

In the case of a multi-user environment, it is necessary to inform the DTM about its current access rights to its instance data especially if another DTM gets the write access to the instance data. See 8.5 for more information on scenarios for multiple users. This service informs a DTM that another instance of the DTM has locked the data. The arguments for the service are shown in Table 49.

Table 49 – Arguments for service OnLockInstanceData

	Request	Response
fdt:SystemTag	M	–
user:UserName	M	–

If another DTM has locked instance data via service LockInstanceData (see 7.7.6.1) the Frame Application has to send OnLockInstanceData to all DTM instances that have a reference to the same instance data.

Receiving this notification, a DTM shall not allow any operations to modify the instance related data, for example by disabling its input fields in the case of an open user interface.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.13.2 Service OnUnlockInstanceData

In the case of a multi-user environment, it is necessary to inform a DTM about its current access mode especially if another DTM gets the read/write access for its instance related data. This service informs a DTM that another DTM has released the lock on the instance data. The arguments for the service are shown in Table 50.

Table 50 – Arguments for service OnUnlockInstanceData

	Request	Response
fdt:SystemTag	M	–
user:UserName	M	–
ServiceSucceeded	–	M

If a DTM has unlocked an instance data via the service UnlockInstanceData (see 7.7.6.2) the Frame Application has to send information via OnUnlockInstanceData to all DTMs which have a reference to the same instance data. When receiving this notification, a DTM supporting data synchronization returns a positive response and shall allow alternation of instance data. DTMs which do not support data synchronization shall return negative response value and shall not allow any data modifications. See 8.5.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.13.3 Service OnInstanceDataChanged

In the case of a multi-user environment, it is necessary to inform the DTM about data changes. This service informs a DTM about changes on the instance data. The arguments for the service are shown in Table 51.

Table 51 – Arguments for service OnInstanceDataChanged

	Request	Response
fdt:SystemTag	M	–
Information	M	–

If a DTM has changed and stored data, it has to call the FA service InstanceDataChanged (see 7.7.6.3) with information of the changed data. The Frame Application will forward it to all other DTMs that have a reference to the same instance data and support synchronized locking by calling this DTM service OnInstanceDataChanged.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.13.4 Service OnChildInstanceDataChanged

In the case of an FDT topology, it is necessary to inform the Parent DTM about the changed data. This service informs a DTM about changes to instance data of a Child DTM. The arguments for the service are shown in Table 52.

Table 52 – Arguments for service OnInstanceChildDataChanged

	Request	Response
fdt:SystemTag	M	–

If a DTM has changed any data, it has to call the FA service InstanceDataChanged (see 7.7.6.3) with information containing the instance-specific changes. The Frame Application will forward this document by calling the DTM service OnInstanceDataChanged (see 7.2.13.3) from all DTMs which have reference to the same instance data.

In regard to the corresponding Parent DTMs (primary parent, secondary parents (see service GetParentNodes, 7.7.3.5), the Frame Application shall implement the following behavior: the Frame Application shall send a notification to the corresponding Parent DTM via this DTM service OnChildInstanceDataChanged (see 7.2.13.4) within a multi-user environment, and this notification will only be sent to one Parent DTM; this notification shall be sent to the Parent DTM which has the lock concerning the related instance data.

If currently no Parent DTM is started, the Frame Application shall start the Parent DTM.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.2.14 DTM services related to import and export

7.2.14.1 Service Export

This service enables to build an export image of complete DTM persistent data. The arguments for the service are shown in Table 53.

Table 53 – Arguments for service Export

	Request	Response
fdt:StorageObject	–	M

This service enables to build an export image of complete DTM persistent data, independently of whether it is stored in the Frame Application Project storage or in a private DTM storage (see 4.9). The data returned by the service shall include DTM instance-related and non-instance-related data.

This service is available in states:
[X] 'configured';
[_] 'communication allowed'.

7.2.14.2 Service Import

This service enables to import data earlier exported by service Export (see 7.2.14.1) back into the DTM. The arguments for the service are shown in Table 54.

Table 54 – Arguments for service Import

	Request	Response
fdt:StorageObject	M	–

This service enables to import data exported by service Export (see 7.2.14.1) back into the DTM.

This service is available in states:
[X] 'configured';
[_] 'communication allowed'.

7.3 Presentation object services

Interaction and state control between Frame Application, Presentation object and DTM are technology dependent and defined within an object model integration profile document.

7.4 Channel object service

7.4.1 Channel object service overview

Subclause 7.4 defines the basic channel services which are common for Process and Communication Channels.

7.4.2 Service ReadChannelInformation

This service returns general (protocol-independent) channel information. The arguments for the service are shown in Table 55.

Table 55 – Arguments for service ReadChannelInformation

	Request	Response
fdt:Tag		M
fdt:ChannelPath		M
fdt:ChannelId		M
fdt:FrameApplicationTag		O

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.4.3 Service WriteChannelInformation

This service writes general (protocol-independent) channel information. The arguments for the service are shown in Table 56.

Table 56 – Arguments for service WriteChannelInformation

	Request	Response
fdt:ProtectedByChannelAssignment	M	–
fdt:FrameApplicationTag	O	–
fdt:serviceSucceeded		M

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.5 Process Channel object services – services for I/O related information

7.5.1 Service ReadChannelData

This service returns information with fieldbus-dependent channel parameters. The arguments for the service are shown in Table 57.

Table 57 – Arguments for service ReadChannelData

	Request	Response
fdt:ProtocolId	M	
fieldbus:ChannelData		M

The service returns a description with the channel-specific parameters. The fieldbus is selected by the parameter ProtocolId.

The caller of service ReadChannelData should use the ProtocolId from the member NetworkInfo in the DtmDeviceInstanceTopology information available via the service GetActiveTypeInfo.

The returned parameters are especially used for channel assignment.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.5.2 Service WriteChannelData

This service sets changes of channel-specific parameters. The arguments for the service are shown in Table 58.

Table 58 – Arguments for service WriteChannelData

	Request	Response
fdt:ProtocolId	M	
fieldbus:ChannelData	M	
fdt:serviceSucceeded		M

The service receives the channel-specific parameters according to a fieldbus-specific description. The fieldbus is defined by the parameter ProtocolId.

The service returns if the channel has accepted the complete information with all changes, or else the channel has rejected all transferred changes. In this case, the channel informs the Frame Application about the error in detail via service OnErrorMessage (see 7.7.2.1).

The service works on the transient data of a DTM. That means that the new data is not stored implicitly.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.6 Communication Channel object services

7.6.1 Services related to communication

7.6.1.1 Service GetSupportedProtocols

This service returns information about the supported protocols of the Communication Channel. The arguments for the service are shown in Table 59.

Table 59 – Arguments for service GetSupportedProtocols

	Request	Response
fdt:BusCategory List	–	M

The service returns fieldbus protocol UUIDs. The supported protocols may be static or depend on the device configuration. If a channel supports more than one protocol during runtime, it has to support all protocols in parallel.

The service is not allowed to change the supported protocols if a DTM is linked to the channel because a change may cause an invalid topology (see 4.3). Which protocols are supported may be determined during topology planning. So, if the Communication Channel can be configured to support different protocols, this is only allowed if there are no linked DTMs.

This service is available in states:
[_] 'configured';
[X] 'communication allowed'.

7.6.1.2 Service Connect

This service establishes a new communication link to a device. The arguments for the service are shown in Table 60.

Table 60 – Arguments for service Connect

	Request	Response
fdt:CommunicationClientObjectReference	M	–
fdt:SystemTag	O	–
fdt:BusCategory	M	–
fieldbus:ConnectRequest	M	–
fieldbus:ConnectResponse	–	M
fdt:CommunicationReferenceID	–	M

The service takes information with fieldbus-specific communication parameters. The fieldbus protocol to be used is identified by the parameter BusCategory.

The request contains additional fieldbus-protocol-specific information for the Communication Channel on how to establish the connection, such as repeat counts or preamble counts, etc. It is up to the environment to decide whether this information fits.

Furthermore, the request contains fieldbus- and protocol-specific information on how to address the device connected to a specific fieldbus.

The SystemTag (see Table A.4) provided in the connect request is the SystemTag of the communication client. It may be used to get access to the DTM by calling the service GetDtm (see 7.7.3.7). If the SystemTag is not provided, then the communication client is not a DTM (it may be the Frame Application or some other component).

This service is available in states:
[_] 'configured';
[X] 'communication allowed'.

7.6.1.3 Service Disconnect

This service releases a communication link to a device. For more than one connection to the same device, the link is identified by the communication reference returned by service Connect (see 7.6.1.2). The arguments for the service are shown in Table 61.

Table 61 – Arguments for service Disconnect

	Request	Response
fdt:CommunicationReferenceID	M	–
fieldbus:DisConnectRequest	M	–
fieldbus:DisConnectResponse	–	M

Via this service the caller of the service receives from the Communication Channel the information as to whether the connection is released.

The service causes the release of all pending response data and at least the release of the CommunicationClientObjectReference passed by service Connect (see 7.6.1.2).

This service is available in states:
[_] 'configured';
[X] 'communication allowed'.

7.6.1.4 Service AbortRequest

This service aborts a communication link to a device without any response. The arguments for the service are shown in Table 62.

Table 62 – Arguments for service AbortRequest

	Request	Response
fdt:CommunicationReferenceID	M	–

The service terminates all pending requests and returns without waiting for a result. The termination of the connection will not be confirmed.

This service is available in states:
[_] 'configured';
[X] 'communication allowed'.

7.6.1.5 Notification service AbortIndication

This service provides a notification that a connection (identified by the CommunicationReferenceID) has been aborted. The arguments for the service are shown in Table 63.

Table 63 – Arguments for service AbortIndication

	Request	Response
fdt:CommunicationReferenceID		M

The service returns the abort notification to a connected communication component or to a connected DTM, that a connection is terminated. All pending requests on this connection are also terminated.

The termination of the connection will not be confirmed.

A termination of a connection can be the result of an invoked service or can occur "spontaneously" (e.g. if the absence of a device is noted automatically by the underlying communication infrastructure).

This service is available in states:
[_] 'configured';
[X] 'communication allowed'.

7.6.1.6 Service Transaction

The service performs exchange of a data structure with a device specified by the communication client. The arguments for the service are shown in Table 64.

Table 64 – Arguments for service Transaction

	Request	Response
fdt:CommunicationReferenceID	M	–
fieldbus:TransactionRequest	M	–
fieldbus:TransactionReponse	–	M

The service returns protocol-specific communication responses for protocol-specific communication requests.

Developers of Communication Channels should not expect that there will be only one pending request for a certain device at one time. For instance, several clients (e.g. Frame Application and DeviceDTMs) are trying to retrieve information from the same device.

Even if the underlying fieldbus protocol allows sending only one request at a time to one or more devices, Communication Channels shall be able to manage a number of requests.

It is expected that the requests are processed by the Communication Channel in the order received if not specified otherwise by the protocol.

Developers of DeviceDTMs should consider that the used communication infrastructure creates delays in the communication. So the DeviceDTMs should limit the number of communication requests.

Also the DeviceDTM shall be able to handle a refused request, since there may be a variety of reasons to refuse a transaction request.

This service is available in states:
[_] 'configured';
[X] 'communication allowed'.

7.6.1.7 Service SequenceDefine

The service SequenceDefine defines a sequence of communication transactions and prepares them for execution. The arguments for the service are shown in Table 65. The communication transactions will be completed when the SequenceStart service is requested.

There is only one sequence defined from a DTM and all transactions requested by the previous sequences are terminated.

Table 65 – Arguments for service SequenceDefine

	Request	Response
fdt:CommunicationReferenceID	M	
fieldbus:SequenceDefine	M	–
fdt:serviceSucceeded		M

The Communication Channel has to preserve the defined sequence until SequenceStart service is requested.

This service is available in states:
[_] 'configured';
[X] 'communication allowed'.

7.6.1.8 Service SequenceStart

The SequenceStart service completes the execution of the sequence of communication transactions defined by the SequenceDefine service (see 7.6.1.7). The arguments for the service are shown in Table 66.

Table 66 – Arguments for service SequenceStart

	Request	Response
fdt:CommunicationReferenceID	M	
fdt:serviceSucceeded		M

The sequence of communication transactions is completed in the sequence the transactions are defined without interruptions until the last transaction service is executed. The Communication Channel informs the client of the result of each transaction when it is completed.

This service is available in states:
[_] 'configured';
[X] 'communication allowed'.

7.6.2 Services related to sub-topology management

7.6.2.1 Service ValidateAddChild

The service validates whether a given DTM, specified by its SystemTag, can be added to the sub-topology (see 4.3). The arguments for the service are shown in Table 67.

Table 67 – Arguments for service ValidateAddChild

	Request	Response
fdt:SystemTag	M	–
fdt:serviceSucceeded	–	M

The channel validates the link. If the link is valid it will return success.

If the DTM needs more information about the child, it is able to access the DTM via the service GetDtm (see 7.7.3.7) passing the received SystemTag.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.6.2.2 Service ChildAdded

The channel is informed that a new DTM was added to the sub-topology. The arguments for the service are shown in Table 68.

Table 68 – Arguments for service ChildAdded

	Request	Response
fdt:SystemTag	M	–

The channel is informed that a new DTM, specified by its SystemTag, has been added to the sub-topology.

If the channel needs more information about the Child DTM, it is able to access the DTM via service GetDtm (see 7.7.3.7) passing the received SystemTag.

This service shall only be used in order to inform that the topology was changed. In case of reloading, for example project related data, this service shall not be called.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.6.2.3 Service ValidateRemoveChild

This service validates if a given Child DTM, specified by its SystemTag, can be removed from the sub-topology. The arguments for the service are shown in Table 69.

Table 69 – Arguments for service ValidateRemoveChild

	Request	Response
fdt:SystemTag	M	–
fdt:serviceSucceeded	–	M

The channel has to validate if the DTM can be removed from the topology.

If the DTM needs more information about the Child DTM, it is able to access the DTM via the service GetDtm (see 7.7.3.7) passing the received SystemTag.

The validation shall include a check for active connection and return failure if a connection is active via this channel.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.6.2.4 Service ChildRemoved

With this service the channel is informed that a linked DTM was removed from the sub-topology. The arguments for the service are shown in Table 70.

Table 70 – Arguments for service ChildRemoved

	Request	Response
fdt:SystemTag	M	–

The channel is informed that a linked DTM, specified by its SystemTag, has been removed from the sub-topology.

If the channel needs more information about the Child DTM, it is able to access the DTM via service GetDtm (see 7.7.3.7) passing the received SystemTag.

This service shall only be used in order to inform that the topology was changed. In the case of un-loading, for example project related data, this service shall not be called.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.6.2.5 Service SetChildrenAddresses

This service requests a bus address setting for a specified device list and returns the device list including success information (see 6.1). The arguments for the service are shown in Table 71.

Table 71 – Arguments for service SetChildrenAddresses

	Request	Response
devList DeviceList	M	–
devList DeviceList	–	M

This service requests the setting of the bus address for one device or for a list of devices. The request may specify that the called Communication Channel shall open a user interface to request device address settings from user. To get a qualified response, error information is included in the returned information.

As part of executing this service, the service OpenPresentationRequest (7.7.8.1) may be used to request address selection from the user.

The bus address on a DTM is set via the service NetworkManagementInfoWrite (see 7.2.11.2). This DTM shall not use this information for execution of communication transactions that set the address in the field device.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.6.3 Services related to GUI and functions

7.6.3.1 Service GetChannelFuntions

This service returns information about functions of a DTM Channel object. The arguments for the service are shown in Table 72.

Table 72 – Arguments for service GetChannelFunctions

	Request	Response
fdt:OperationPhase	M	–
func:Functions	–	M

This service provides information about functions provided by channel.

A channel shall inform the Frame Application about any function changes (e.g. number, status, label, etc.) by calling the service OnFunctionChanged (see 7.7.2.4).

Several responses may be provided. Usually, this information is used by the Frame Application to create menus.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.6.3.2 Service GetGuiInformation

This service provides information how to start the user interface of a channel. The arguments for the service are shown in Table 73.

Table 73 – Arguments for service GetGuiInformation

	Request	Response
func:FDTFunctionCall	M	–
func:GUIInformation	–	M

Returns information that enables the Frame Application to instantiate the user interface. Channels shall only support Presentation objects that allow integration into the GUI of the Frame Application.

Integration and Interaction between these objects is technology dependent and defined within object model integration profile documents.

This service is available in states:
[X] 'configured';
[X] 'communication allowed'.

7.6.4 Service Scan

This service performs the scan to the sub-topology (see 6.2.2). The arguments for the service are shown in Table 74.

Table 74 – Arguments for service Scan

	Request	Response
devList:BusAddressRange	O	–
fieldbus:ScanIdentifications	–	M

This service returns information which contains a list of fieldbus related information to identify the connected devices. The optional passed BusAddressRange may limit the range of the scanning to specific device addresses or ranges.

This service is available in states:
[_] 'configured';
[X] 'communication allowed'.

7.7 Frame Application services

7.7.1 General state availability

All Frame Application services are available in DTM states:
[X] 'configured';
[X] 'communication allowed'.

7.7.2 FA services related to general events

7.7.2.1 Service OnErrorMessage

With this service a Frame Application receives notification by a DTM about errors during a service call. The arguments for the service are shown in Table 75.

Table 75 – Arguments for service OnErrorMessage

	Request	Response
fdt:SystemTag	M	–
fdt:ErrorMessage	M	–

The service is necessary if the DTM works without a user interface. If a DTM works without a user interface it is not allowed to display any error message within its own dialog windows.

It is up to the Frame Application to handle the information. In case of errors or warnings, the human readable string may be displayed in a dialog box of the Frame Application.

In order to provide users with helpful information (e.g. regarding errors), it is recommended to use this service in cases where a service call failed and the service UserDialog (see 7.7.8.3) is not used.

7.7.2.2 Service OnProgress

With this service a Frame Application receives notification by a DTM about the progress on the handling of a service call. The arguments for the service are shown in Table 76.

Table 76 – Arguments for service OnProgress

	Request	Response
fdt:SystemTag	M	–
fdt:ProgressInformation	M	–

This service shall be used by DTMs during active service calls that take a longer time to inform the Frame Application and at least the user about ongoing activities. If a DTM is not able to determine the real progress, it may be useful to change, for example, the title.

It is up to the Frame Application how to handle the information.

7.7.2.3 Service OnOnlineStatusChanged

With this service a Frame Application receives notification by a DTM about the status of the communication link. The arguments for the service are shown in Table 77.

Table 77 – Arguments for service OnOnlineStatusChanged

	Request	Response
fdt:SystemTag	M	–
fdt:OnlineStatus (previous)	M	–
fdt:OnlineStatus (current)	M	
fdt:CommunicationError	O	

This service shall be used by DTMs to inform the Frame Application about changes of the communication link status. The DTM shall specify the new and previous online status and additional error information, if the status change was triggered by a communication error.

7.7.2.4 Service OnFunctionsChanged

With this service a Frame Application receives notification by a DTM that the information about its current available additional functionality has changed. The arguments for the service are shown in Table 78.

Table 78 – Arguments for service OnFunctionsChanged

	Request	Response
fdt:ChannelPath	Ö	–
fdt:SystemTag	M	–

This service shall be used by DTMs to inform the Frame Application to update its menus or function calls which reference on this extended functionality. The Frame Application gets the currently available DTM functions by the service GetFunctions (see 7.2.5.1).

If the parameter ChannelPath is provided, then the functionality of the corresponding channels has changed. In this case, Frame Application gets the currently available channel functions by the service GetChannelFunctions (see 7.6.3.1).

7.7.3 FA services related to topology management

7.7.3.1 Service GetDtmInfoList

This service returns a list of DTM related information. The arguments for the service are shown in Table 79.

Table 79 – Arguments for service GetDtmInfoList

	Request	Response
dtmInfo:DTMInfo list	–	M
dtmInfo.BTMInfo list		M

This service returns a list of DTM related information. This information enables a DTM to add another DTM to the FDT topology by usage of the service CreateChild (see 7.7.3.2).

It is up to the Frame Application to decide which DTM information is returned in the list. For example, the list could contain only information concerning DTMs for one communication protocol even if DTMs for other protocols are installed.

7.7.3.2 Service CreateChild

This service enables a DTM to add another DTM to the FDT topology. The arguments for the service are shown in Table 80.

Table 80 – Arguments for service CreateChild (DTM)

	Request	Response
fdt:DtmDeviceType	M	–
fdt: ChannelObjectReference	M	–
fdt:SystemTag	–	M

This service enables a DTM to add another DTM identified by its DTM Device Type to the sub-topology identified by the ChannelObjectReference.

The DTM is instantiated by the Frame Application. The service returns the SystemTag of the DTM. If the operation failed, no SystemTag shall be returned. The Frame Application has to implement the behavior described in the sequence diagram in 8.1.2.

If the service is called to add a BTM, then the BTM block type is used in the request as defined in Table 81.

Table 81 – Arguments for service CreateChild (BTM)

	Request	Response
btm:BtmBlockType	M	–
fdt: ChannelObjectReference	M	–
fdt:SystemTag	–	M

7.7.3.3 Service DeleteChild

This service enables a DTM to remove another DTM from the FDT topology. The arguments for the service are shown in Table 82.

Table 82 – Arguments for service DeleteChild

	Request	Response
fdt:SystemTag	M	–
fdt: ChannelObjectReference	M	–
fdt:serviceSucceeded	–	M

This service enables a DTM to remove another DTM specified by the SystemTag from the sub-topology identified by the ChannelObjectReference.

The Frame Application has to call the service ValidateRemoveChild (see 7.6.2.3) at the Communication Channel from which the DTM is removed. If this was the last reference within the topology, the Frame Application has to delete the instance data. The Frame Application has also to call the service ClearInstanceData (see 7.2.4.5) with respect to the behavior. The operation shall also fail if a sub-topology exists.

7.7.3.4 Service MoveChild

This service enables a DTM to move another DTM in the FDT topology. The arguments for the service are shown in Table 83.

Table 83 – Arguments for service MoveChild

	Request	Response
fdt:SystemTag	M	–
fdt:SystemTag	M	
fdt: ChannelObjectReference	M	–
fdt:serviceSucceeded	–	M

This service enables a DTM to move another DTM specified by its SystemTag to another sub-topology identified by the destination DTM SystemTag and ChannelObjectReference.

The Frame Application has to call ValidateAddChild and ValidateRemoveChild as defined for the services CreateChild (see 7.7.3.2) and DeleteChild (see 7.7.3.3).

7.7.3.5 Service GetParentNodes

This service enables a DTM to request a list of DTMs which are directly above in the FDT topology. The arguments for the service are shown in Table 84.

Table 84 – Arguments for service GetParentNodes

	Request	Response
fdt:SystemTag	M	–
fdt:SystemTag List	–	M
fdt:SystemTag		M

The DTM for which the list shall be created is identified by the SystemTag in the request. The service returns a SytemTag list of all DTMs above in the FDT topology and a single SystemTag identifying the primary Parent DTM.

7.7.3.6 Service GetChildNodes

This service enables a DTM to request a list of DTMs which are directly below in the FDT topology. The arguments for the service are shown in Table 85.

Table 85 – Arguments for service GetChildNodes

	Request	Response
fdt:SystemTag	M	–
fdt: ChannelObjectReference	M	–
fdt:SystemTag List	–	M

The DTM and Communication Channel for which the list shall be created are identified by the SystemTag and a ChannelObjectReference. The service returns a SytemTag list of all DTMs below in the FDT topology.

7.7.3.7 Service GetDtm

This service enables a DTM to request a reference to another DTM identified by its SystemTag. The arguments for the service are shown in Table 86.

Table 86 – Arguments for service GetDtm

	Request	Response
fdt:SystemTag	M	–
fdt: DtmObjectReference	–	M

This service enables a DTM to request a reference to another DTM identified by its SystemTag. Additional calls within a FrameApplication instance shall return the identical DTM ObjectReference.

The DTM which has requested the references has to call the service ReleaseDtm (see 7.7.3.8) to release the reference. The Frame Application has to implement a kind of reference counting which is required to handle multiple references to the same DTM instance.

7.7.3.8 Service ReleaseDtm

This service shall be used to release the reference to the associated DTM that was requested by the service GetDtm (see 7.7.3.7). The arguments for the service are shown in Table 87.

Table 87 – Arguments for service ReleaseDtm

	Request	Response
fdt:SystemTag	M	–
fdt:serviceSucceeded	–	M

This service shall be used to release the reference to the associated DTM, identified by its SystemTag, which was requested by the service GetDtm (see 7.7.3.7).

7.7.4 FA services related to redundancy

7.7.4.1 Service OnAddedRedundantChild

A Parent DTM shall send this information to the Frame Application if another DTM handling a redundant device is added to the topology. The Frame Application is then able to display the instance at an additional redundant Communication Channel. The arguments for the service are shown in Table 88.

Table 88 – Arguments for service OnAddedRedundantChild

	Request	Response
fdt:SystemTag	M	–
fdt:ChannelObjectReference	M	–
fdt:serviceSucceeded	–	M

The Parent DTM has added the DTM identified by its SystemTag, handling a redundant slave, to the Communication Channel identified by ChannelObjectReference. The Frame Application shall not call the service ValidateAddChild (see 7.6.2.1) or ChildAdded (see 7.6.2.2) on this channel.

7.7.4.2 Service OnRemovedRedundantChild

A Parent DTM shall send this information to the Frame Application if another DTM handling a redundant device is removed from the topology. The Frame Application is able to hide the instance at the additional redundant Communication Channel. The arguments for the service are shown in Table 89.

Table 89 – Arguments for service OnRemovedRedundantChild

	Request	Response
fdt:SystemTag	M	–
fdt:ChannelObjectReference	M	–
fdt:serviceSucceeded	–	M

The Parent DTM removes the DTM identified by its SystemTag, handling a redundant slave, from the Communication Channel identified by ChannelObjectReference. The Frame Application shall not call the service ValidateRemoveChild (see 7.6.2.3) or the service ChildRemoved (see 7.6.2.4) on this channel.

7.7.5 FA services related to storage of DTM data

7.7.5.1 Service SaveInstanceData

This service enables a DTM to save its instance data into the Project storage managed by the Frame Application (see 4.9). The arguments for the service are shown in Table 90.

Table 90 – Arguments for service SaveInstanceData

	Request	Response
fdt:StorageObject	M	–
fdt:serviceSucceeded	–	M

It is up to the Frame Application to decide whether data passed in the requests is immediately stored in the Project storage or cached in the memory until another event occurs (e.g. explicit user request).

The DTM calling this service shall hold the lock to the instance data (see service LockInstanceData, 7.7.6.1), otherwise the service call fails (returns fdtServiceSucceeded = FALSE).

7.7.5.2 Service LoadInstanceData

This service enables a DTM to load instance data from the Project storage managed by the Frame Application (see 4.9). The arguments for the service are shown in Table 91.

Table 91 – Arguments for service LoadInstanceData

	Request	Response
fdt:StorageObject	–	M

The DTM calling this service gets exactly the same data which was stored by service SaveInstanceData (see 7.7.5.1).

7.7.5.3 Service GetPrivateDtmStorageInformation

This service enables a DTM to request information about its private data storage from the Frame Application (see 4.9). The arguments for the service are shown in Table 92.

Table 92 – Arguments for service GetPrivateDtmStorageInformation

	Request	Response
InstanceRelatedStorageInfo	–	M
ProjectRelatedStorageInfo	–	M

The response parameter InstanceRelatedStorageInfo contains the information which enables a DTM to access the storage that belongs to the calling instance. The response parameter ProjectRelatedStorageInfo contains the information which enables to access the storage shared by all instances of this DTM type in a Project. The private DTM storage mechanism is implementation technology dependent and thus defined in the object model integration profile document of the IEC 62453 series defining mapping to specific technologies.

If a Frame Application is not able to provide access to private DTM data storages then it returns no information. A DTM shall be able to work without any side effects in this case. It shall ensure that there is no impact to the instance data managed by SaveInstanceData (see 7.7.5.1) and LoadInstanceData (see 7.7.5.2).

A DTM shall clean up the instance related storage if the service ClearInstanceData (see 7.2.4.5) is called. If the DTM holds any references between the Project and instance related storages, then it also shall clean up these in the Project related storage.

7.7.6 FA services related to DTM data synchronization

7.7.6.1 Service LockInstanceData

This service enables a DTM to request an exclusive write access to its instance data. The arguments for the service are shown in Table 93.

Table 93 – Arguments for service LockInstanceData

	Request	Response
fdt:serviceSucceeded	–	M

The Frame Application validates the request and grants the access within the multi-user environment. In single user systems, the exclusive access is always granted. The DTM shall make modifications to instance related data only when access is granted via this service. See 8.5.

The DTM shall request the lock for instance data only when required for data modification purposes. The DTM shall possess an exclusive write access when allowing:

- the configuration of instance data via Presentation objects;
- the synchronization of the data between device and instance data via service ReadDataFromDevice (see 7.2.12.4);
- the device data modification via service DeviceDataWrite (see 7.2.10.4), or via parameterization GUI, to update the information data modified in device (ModifiedInDevice).

If the write access is not granted, the DTM shall not make modifications into instance data and shall inform the user via the service OnErrorMessage (see 7.7.2.1) notification service.

7.7.6.2 Service UnlockInstanceData

This service enables the DTM to notify the Frame Application that it wants to unlock the write access to the instance data. The arguments for the service are shown in Table 94.

Table 94 – Arguments for service UnlockInstanceData

	Request	Response
fdt:serviceSucceeded	–	M

When instance data is not further under modification, the DTM shall immediately save it and call this service. Within the multi-user environment, the Frame Application shall notify other DTM instances having a reference to the same instance data about the changed data via the service InstanceDataChanged (see 7.7.6.3).

If the service response is unsuccessful, the DTM shall inform the user via the service OnErrorMessage (see 7.7.2.1) notification service to clean up the system database. Within the single user systems, the service response is always successful.

7.7.6.3 Service OnInstanceDataChanged

This service enables a DTM to inform the Frame Application about data changes. The arguments for the service are shown in Table 95.

Table 95 – Arguments for service OnInstanceDataChanged

	Request	Response
Information	M	–

If a DTM has changed and saved data, it has to call this service with information on the changed data. The Frame Application shall forward this information to all DTM instances that have a reference to the same instance data by calling the DTM service OnInstanceDataChanged (see 7.2.13.3).

7.7.7 FA service related to process image validation – service ValidateProcessImage

This service enables a DTM to validate whether a modification of the process image can be applied while the PLC is running. The arguments for the service are shown in Table 96.

Table 96 – Arguments for service ValidateProcessImage

	Request	Response
ProcessImageInfo	M	–

In some automation systems it is a requirement to apply changes to the process image while the PLC is running. This service allows to validate whether a potential change can be applied while the PLC is running.

7.7.8 FA services related to presentation

7.7.8.1 Service OpenPresentationRequest

This service enables a DTM or a channel to request a Presentation object in the user interface of the Frame Application. The arguments for the service are shown in Table 97.

Table 97 – Arguments for service OpenPresentationRequest

	Request	Response
func: FDTFunctionCall	M	–
ui: RunAsModal	O	
fdt:ChannelPath	O	
fdt:invokeID	–	M
fdt: ServiceSucceeded	–	M

This service enables a DTM or a channel to open a presentation, identified by a FDTFunctionCall, in the user interface of the Frame Application. The Frame Application shall use the DTM service GetGuiInformation (see 7.2.5.3) or channel service GetGuiInformation (see 7.6.3.2) to obtain more information about the presentation.

If the parameter RunAsModal is set to TRUE then this service behaves in a modal way: the Frame Application at least has to ensure that all the presentations of the calling DTM are disabled, so that no further user input is possible.

If the parameter ChannelPath is provided, then the presentation of the corresponding channels shall be opened.

The service returns an InvokeID that enables to close the opened presentation by calling the service ClosePresentationRequest.

7.7.8.2 Service ClosePresentationRequest

This service enables a DTM or a channel to close an opened Presentation object. The arguments for the service are shown in Table 98.

Table 98 – Arguments for service ClosePresentationRequest

	Request	Response
fdt:invokeID	M	
fdt: ServiceSucceeded	–	M

This service enables a DTM or a channel to close an opened presentation. The passed parameter InvokeID identifies the presentation that was for example opened by service OpenPresentationRequest (see 7.7.8.1).

7.7.8.3 Service UserDialog

This service enables a DTM to open a user message within the user interface of the Frame Application. The arguments for the service are shown in Table 99.

Table 99 – Arguments for service UserDialog

	Request	Response
ui:UserMessage	M	–
ui:UserMessage	–	M

A DTM shall use this service for standard user messages such as error or information messages, especially if a DTM is not allowed to open a private user dialog (see service PrivateDialogEnabled, 7.2.1.1).

This service returns the selection of the user action or the specified default answer of the message. It is up to the Frame Application to display the user message or to send the default answer.

In case of a distributed system, the Frame Application shall ensure the displaying of the user dialog and the user interface of the DTM at the same workplace.

7.7.9 FA Services related to audit trail – service RecordAuditTrailEvent

This service shall be used by all DTMs to send their audit trail information to the Frame Application. The arguments for the service are shown in Table 100. It is up to the Frame Application to implement the audit trail application itself which records the device-specific information and supplies the user interface.

Table 100 – Arguments for service RecordAuditTrailEvent

	Request	Response
fdt:SystemTag	M	–
audittrail:LogEntry	M	–
audittrail:TransactionInfo	O	–

The notification from a DTM about change of data is to be recorded by the Frame Application's audit trail tool.

The content of LogEntry depends on the DTM. The implementation of the audit trail tool is Frame Application-specific.

audittrail:TransactionInfo with value startTransaction shall be used to request the audit trail for the following actions: for example, configuration or simulation. All calls of this service will be recorded until the record is closed by calling it with endTransaction as value of audittrail:TransactionInfo parameter.

When the record has been closed, the Frame Application may use it for its own specific audit trail functions such as adding comments, etc.

8 FDT dynamic behavior

8.1 Generate FDT topology

8.1.1 FDT topology generation triggered by the Frame Application

The Frame Application is responsible for generating and managing the topology. The sequence (see Figure 38) shows how the Frame Application manages the link between DTMs and Communication Channels.

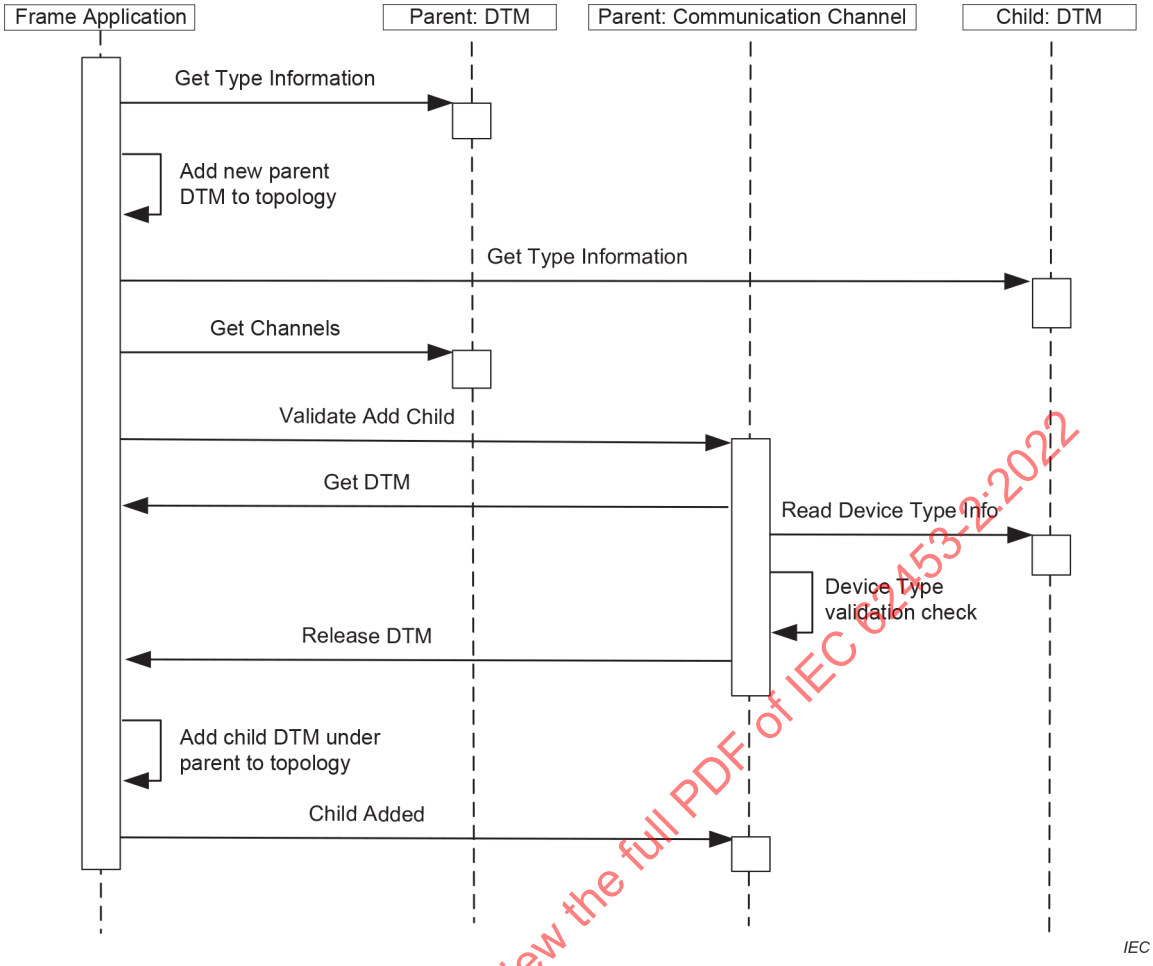


Figure 38 – FDT topology generation triggered by the Frame Applications

8.1.2 FDT topology generation triggered by the DTM

This sequence (see Figure 39) shows the generation of the FDT topology triggered by a DTM.

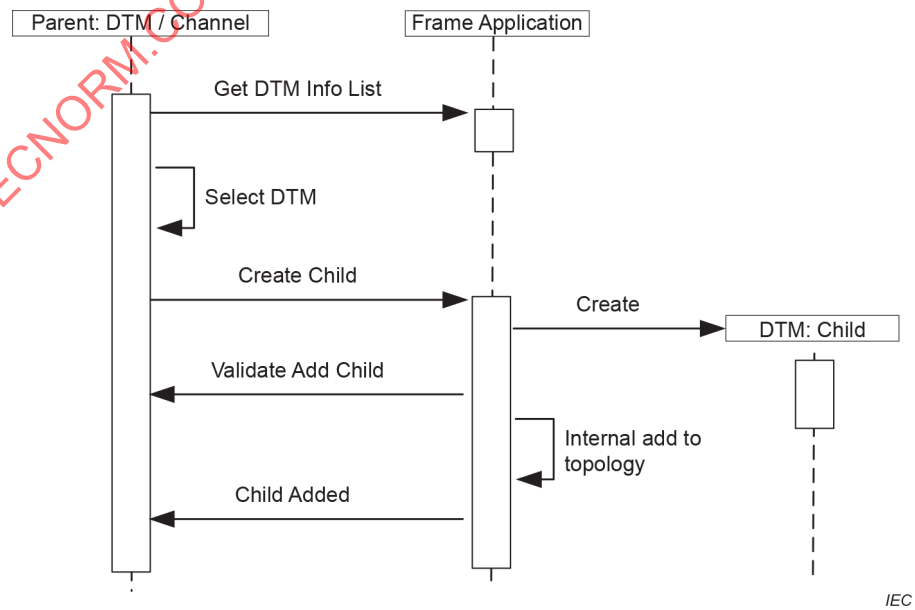


Figure 39 – FDT topology generation triggered by a DTM

8.2 Address setting

8.2.1 Address setting overview

Subclause 8.2 describes different address setting scenarios (see also 6.1).

8.2.2 Set or modify device address – with user interface

In this scenario, the Frame Application requests a set of specific child device addresses at the DTMs. This sequence (see Figure 40) is for example started if a new DTM is added to the topology. A similar sequence can be used if a Frame Application offers a manual change of address in the context of a DTM.

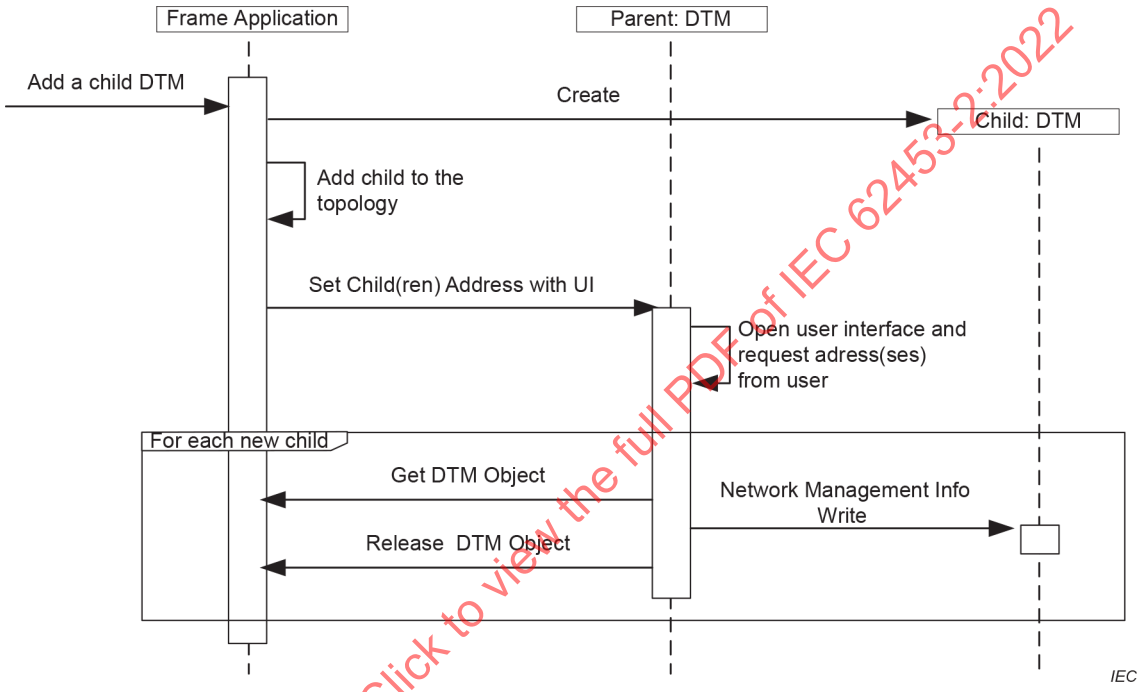


Figure 40 – Set or modify device address – with user interface

8.2.3 Set or modify device address – without user interface

In this scenario, the Frame Application requests to set the device addresses at the DTMs, for example after a scanning or import operation (see Figure 41).

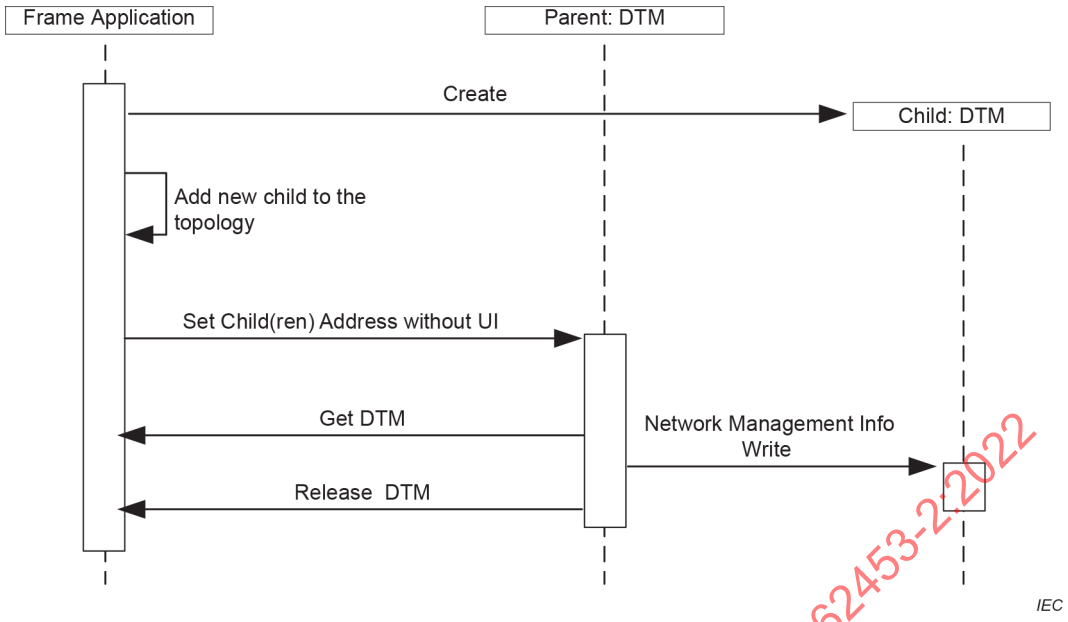


Figure 41 – Set or modify device address – without user interface

The Frame Application submits the list of addresses to the Parent DTM through the service NetworkManagementInfoWrite (see 7.2.11.2).

8.2.4 Display or modify all child device addresses with user interface

In this scenario (see Figure 42) the Frame Application requests the display or modification of all child device addresses at a DTM. This sequence is for example started if the user selects the corresponding menu in the context of a Communication or Gateway DTM.

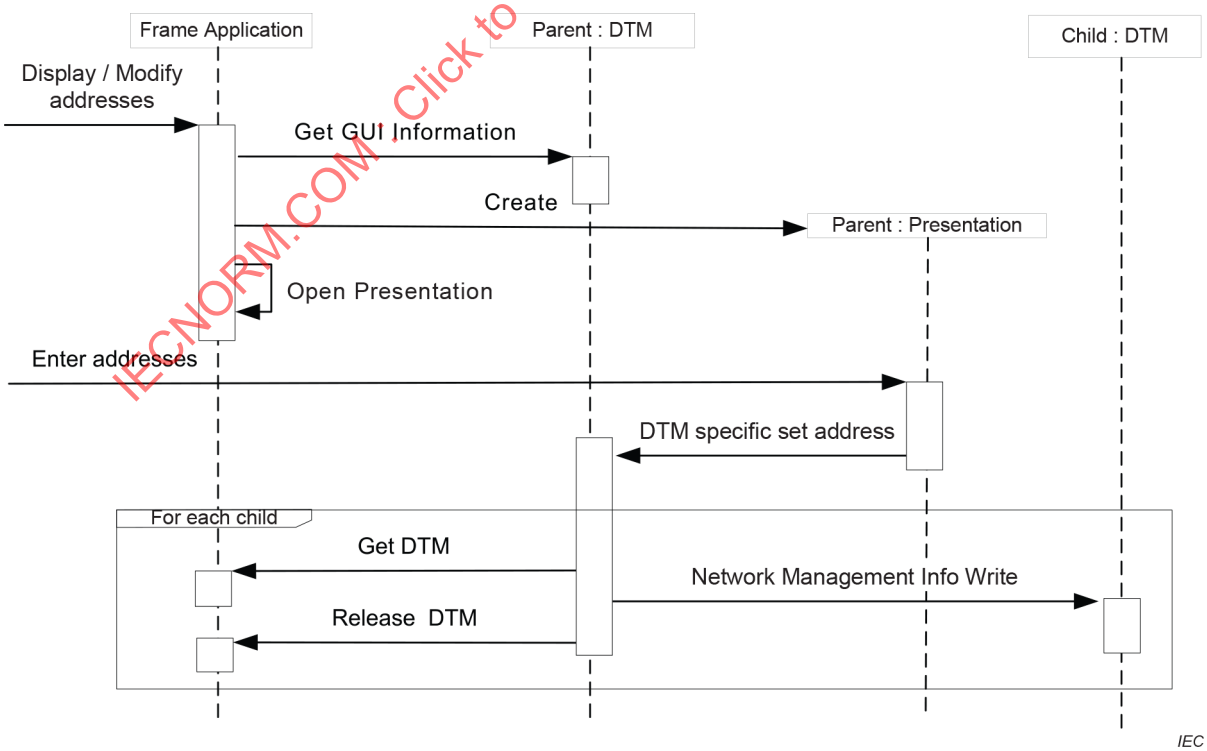


Figure 42 – Set or modify all device addresses – with user interface

8.3 Communication

8.3.1 Communication overview

Subclause 8.3 describes different communication scenarios of a DTM with its device.

8.3.2 Point-to-point communication

This sequence diagram (see Figure 43) shows the point-to-point communication of a DTM by using the services provided by a linked Communication Channel.

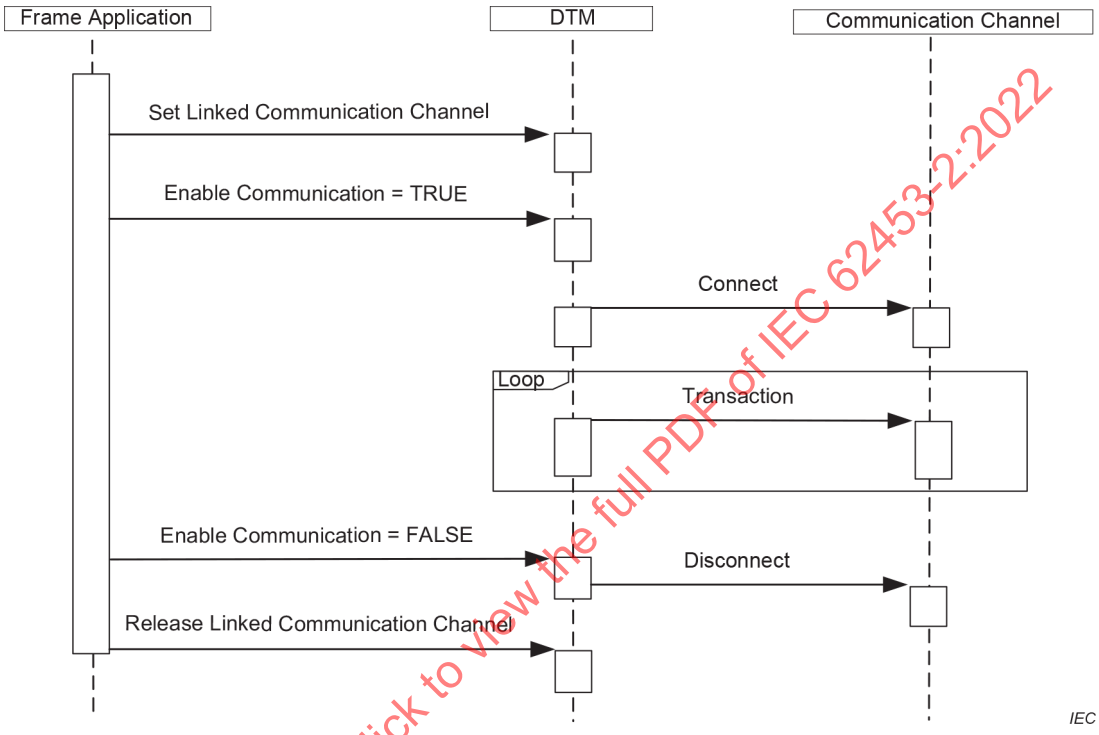


Figure 43 – Point-to-point communication

8.3.3 Nested communication

This sequence diagram (see Figure 44) shows the communication of a DTM through a gateway and a Communication Channel provided by the Frame Application.

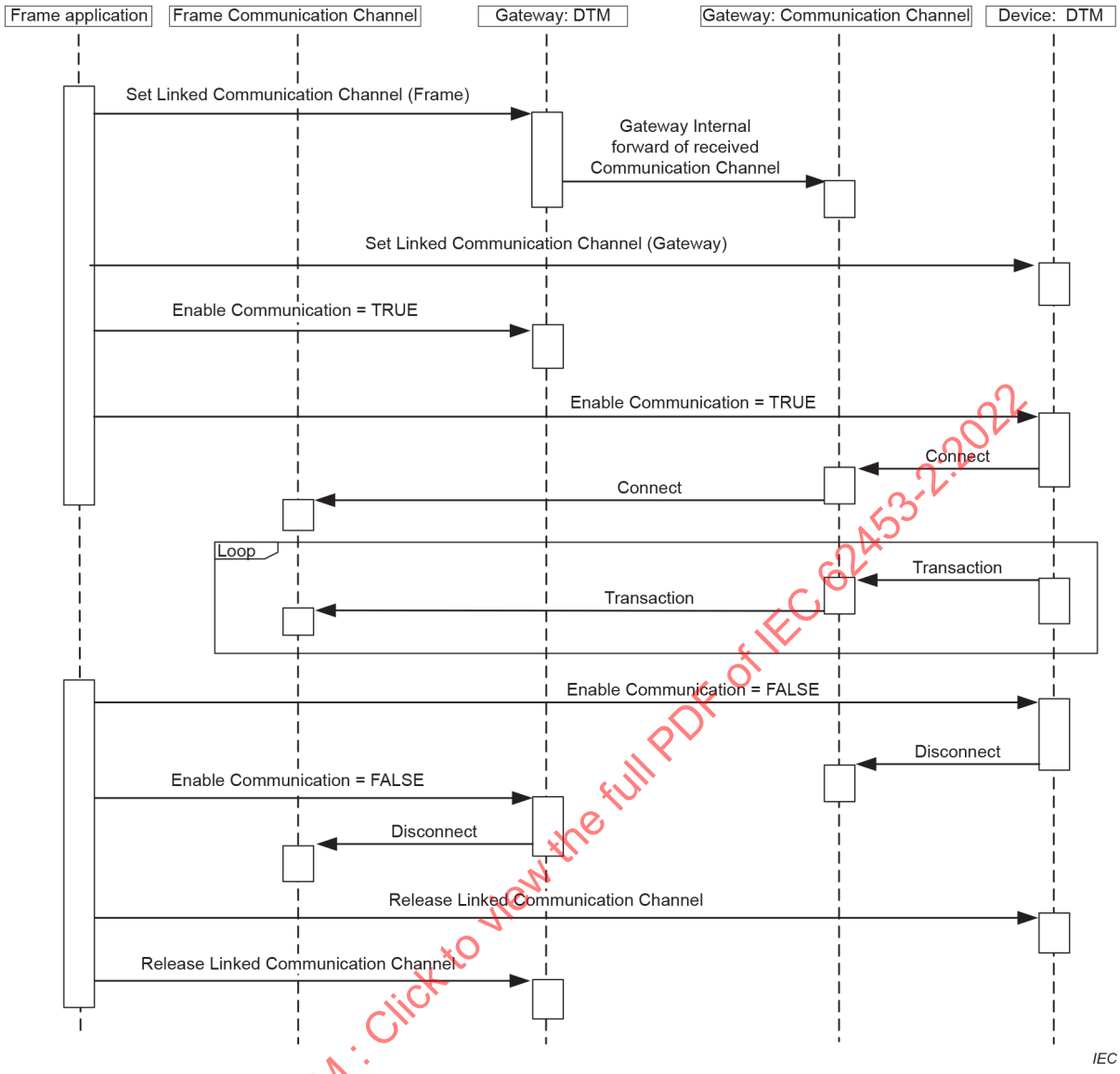


Figure 44 – Nested communication

8.3.4 Device-initiated data transfer

Some protocols support data transfer services that are initiated by the device and not by the DTM. In order to support such services, the following general behavior is defined in FDT:

- the infrastructure (filter, service queue) for such services is initiated by a protocol-specific transaction request of the DTM; depending on the protocol this service may be acknowledged or not;
- the device-initiated data transfer is transported by multiple transaction responses to a single initiating transaction request;
- the infrastructure for these services is terminated by a protocol-specific transaction request of the DTM.

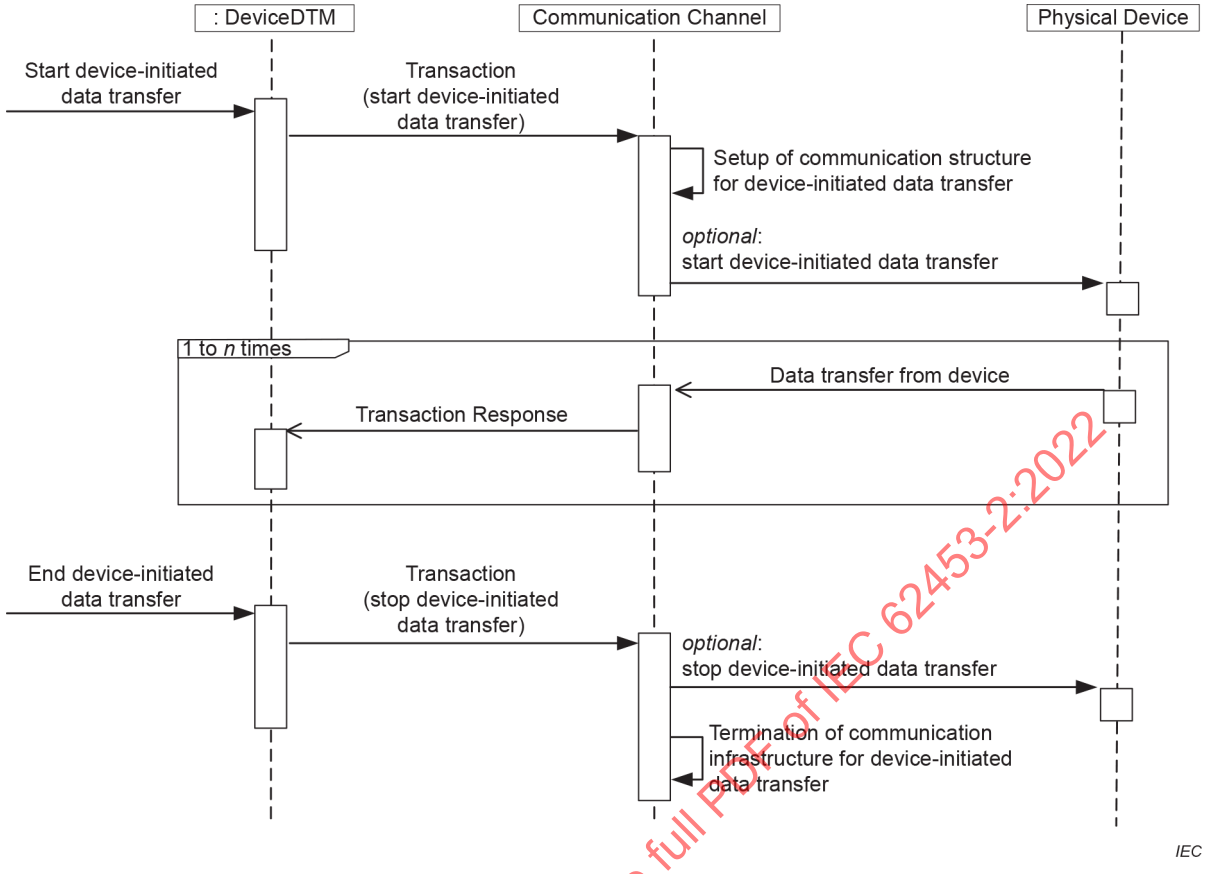


Figure 45 – Device-initiated data transfer

Device-initiated data transfer needs management by the communication infrastructure. That is why it is necessary to unambiguously identify the request to initialize and terminate this type of behavior.

The transaction service (see 7.6.1.6) to initiate or terminate the device-initiated data transfer is transport protocol-specific. The transaction service to terminate the data transfer contains all the information necessary to terminate the device-initiated data transfer (if necessary it also references the transaction service which started the data transfer).

If a DTM starts the device-initiated data transfer service on the device (see Figure 45), it shall also terminate this service. The termination has to occur before the DTM goes offline (see service Disconnect (7.6.1.3) or AbortRequest (7.6.1.4)). If the connection is terminated by the notification AbortIndication (see 7.6.1.5), this service is terminated automatically.

The support of such services is protocol-specific and device-specific. If a Communication Channel is not able to support this type of service, it shall return an error indicating that it is not able to support this feature.

The concrete protocol-specific transactions are defined in the IEC 62453-3xy publications defining the protocol profile integration in FDT.

8.4 Scanning and DTM assignment

This sequence (see Figure 46) shows how a Frame Application generates an FDT sub-topology based on information returned by the service Scan (see 7.6.4).

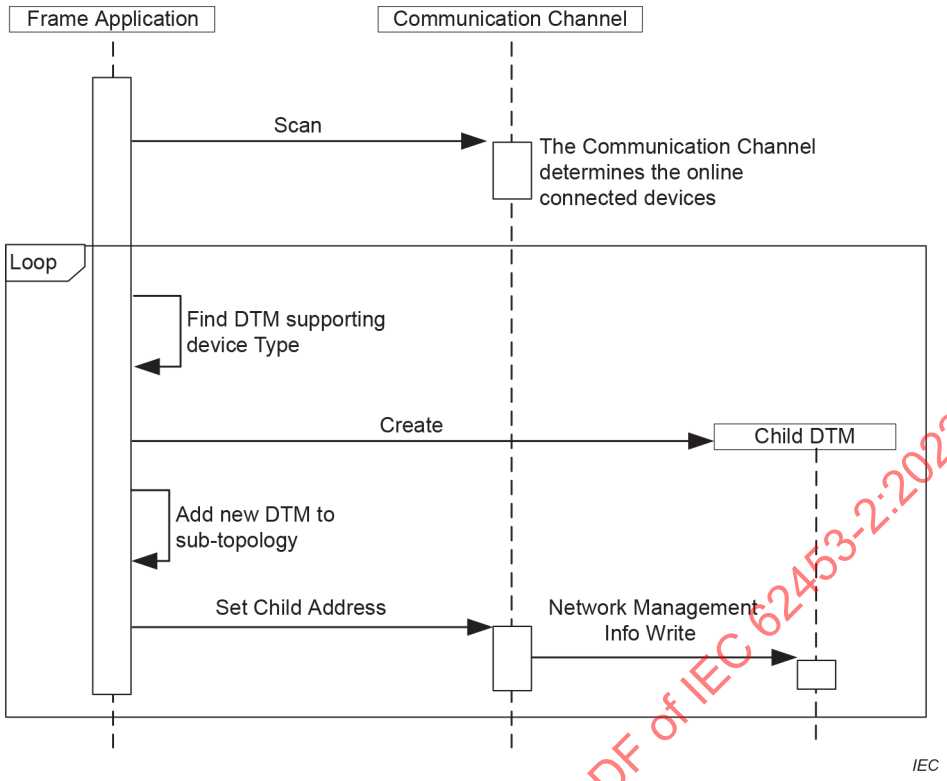


Figure 46 – Scanning and DTM assignment

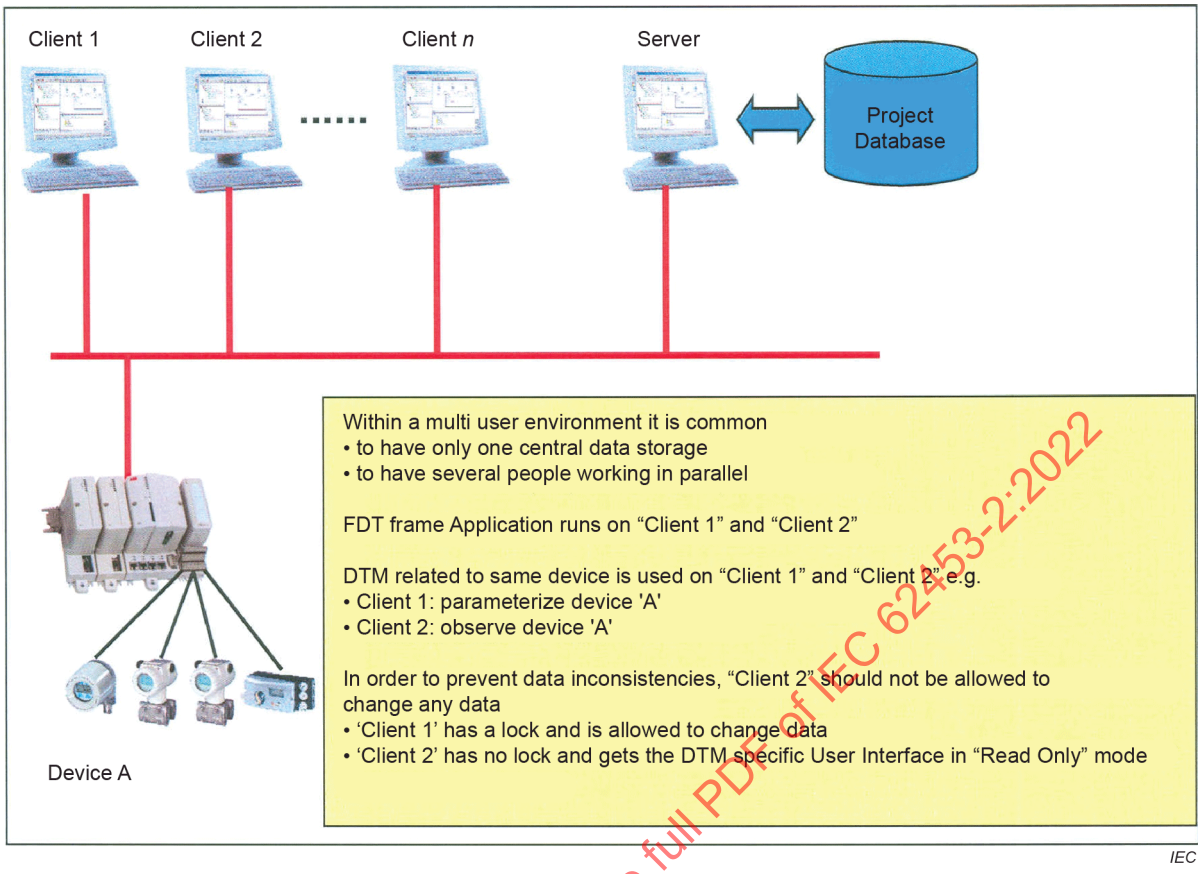
The Frame Application is able to request device type and instance related information by calling the service Scan (see 7.6.4), use the information to find proper DTMs that can be added to the FDT topology (see 6.2.3, DTM assignment) and finally set the addresses in the DTMs to enable them to communicate with its device (see 6.1, address management).

8.5 Multi-user scenarios

8.5.1 General

In order to reduce time, for example the commissioning time of a large installation, systems are often designed for multi-user access. The goal is to allow a number of users to work in parallel and ensure data consistency, while preventing unintentional change of data, for example configuration related data based on parallel work. Figure 47 gives an overview of a multi-user system.

The mechanism to handle changes within the device (for example local operation on the device) is described in 8.7.4.



NOTE This figure shows one Frame Application managing multiple users (e.g. multiple instances of one Frame Application running on several clients working on the same data base). Multiple Frame Applications for multiple users is outside the scope of this document.

Figure 47 – Multi-user system

Within a multi-user environment, it is common that more than one DTM has access to the instance data. To synchronize DTMs which are started by several users on different workplaces, FDT provides a locking mechanism. The target for this event mechanism is that only one DTM has read/write access to instance related data. All other DTMs have only a read access.

Access to the stored data set shall be managed during all modifications of the data set. This includes modification via the GUI of the DTM and modification via the InstanceData services of the DTM.

In order to prevent multi-DTM-access to the device, online writable access to a device shall be provided only if the instance data set is locked. This means the data set shall be locked only if services are executed that change the device data (e.g. Online Parameterization GUI and service WriteDataToDevice) but not when device data is read only (e.g., observation).

For this reason, a DTM shall lock its data set only if required and only during modification of the data. After the instance data is saved by the Frame Application and is not further under modification the DTM shall unlock its instance data immediately to enable concurrent access to the device data by other DTM instances within a multi-user environment.

Before opening the GUI for modification of data, the DTM shall try to lock the instance data. If the service is successful all modification to the instance data is allowed, for example parameterization, synchronization. If lock is not granted, the DTM is not allowed to make any alterations to its instance data.

When there is no need to modify the data (e.g. all GUIs with write access are closed, synchronization is completed), the DTM shall request to save the instance data and unlock it when saved.

The DTM is not allowed to hold write access to its instance related data unnecessarily. Otherwise, the DTM is not suitable for use in a multi-user environment. For example, the DTM should not lock the instance data immediately when it is initialized and unlock it only when terminated.

8.5.2 Synchronized and non-synchronized locking mechanism for DTMs

The DTM shall always support a locking mechanism. There are two options: the synchronized and non-synchronized mechanism. The supported mechanism type is informed by the Frame Application in a response of the service OnUnlockInstanceData (see 7.2.13.2).

The following sequence diagram (see Figure 48) shows the general flow of events in the case of a DTM supporting the synchronized locking mechanism.

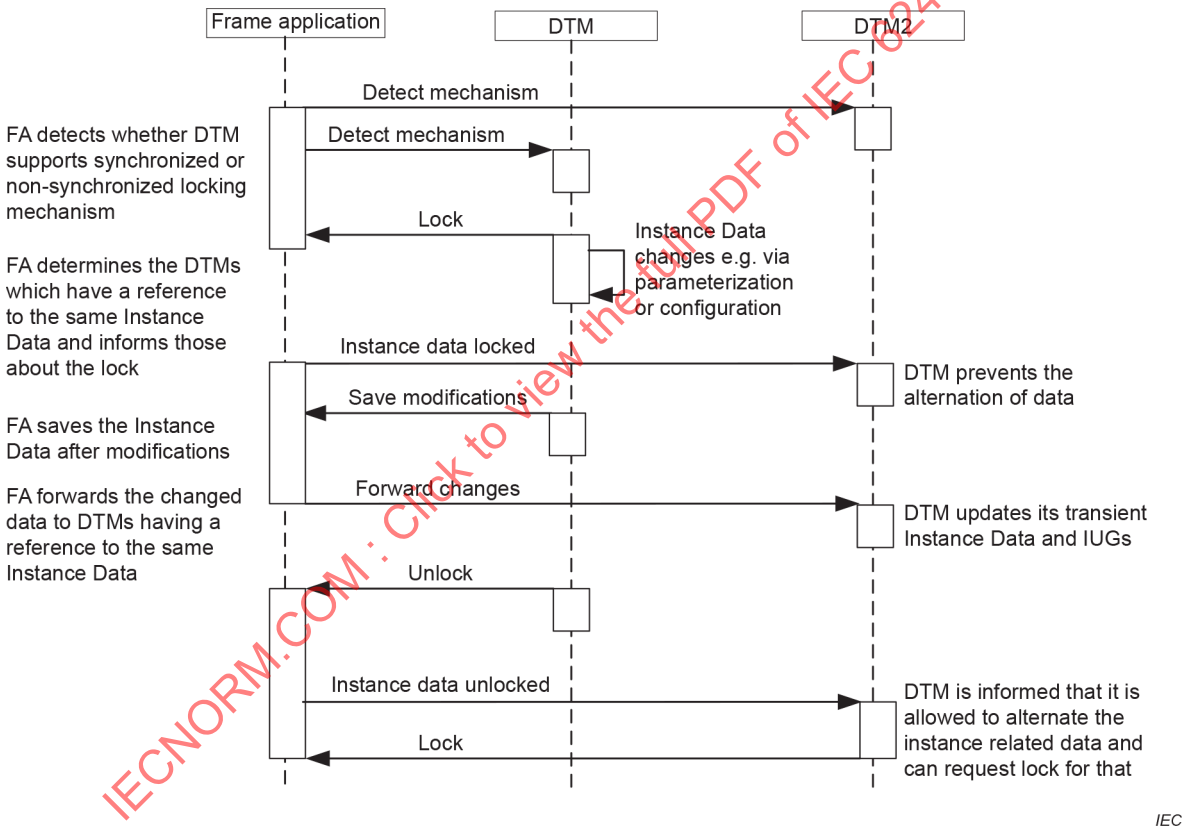


Figure 48 – General synchronized locking mechanism

The following sequence diagram (see Figure 49) shows the general flow of events in the case of a DTM supporting the non-synchronized locking mechanism.

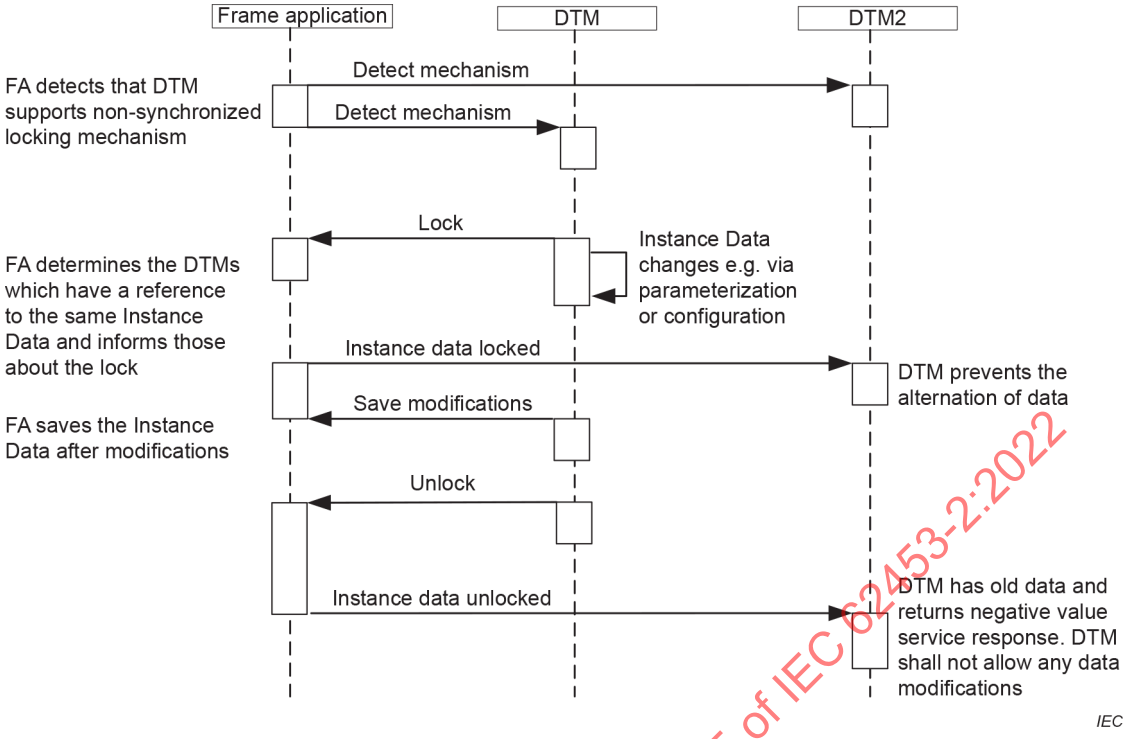


Figure 49 – General non-synchronized locking mechanism

The following sequence diagram (see Figure 50) shows a parameterization scenario in the case of a DTM supporting the synchronized locking mechanism.

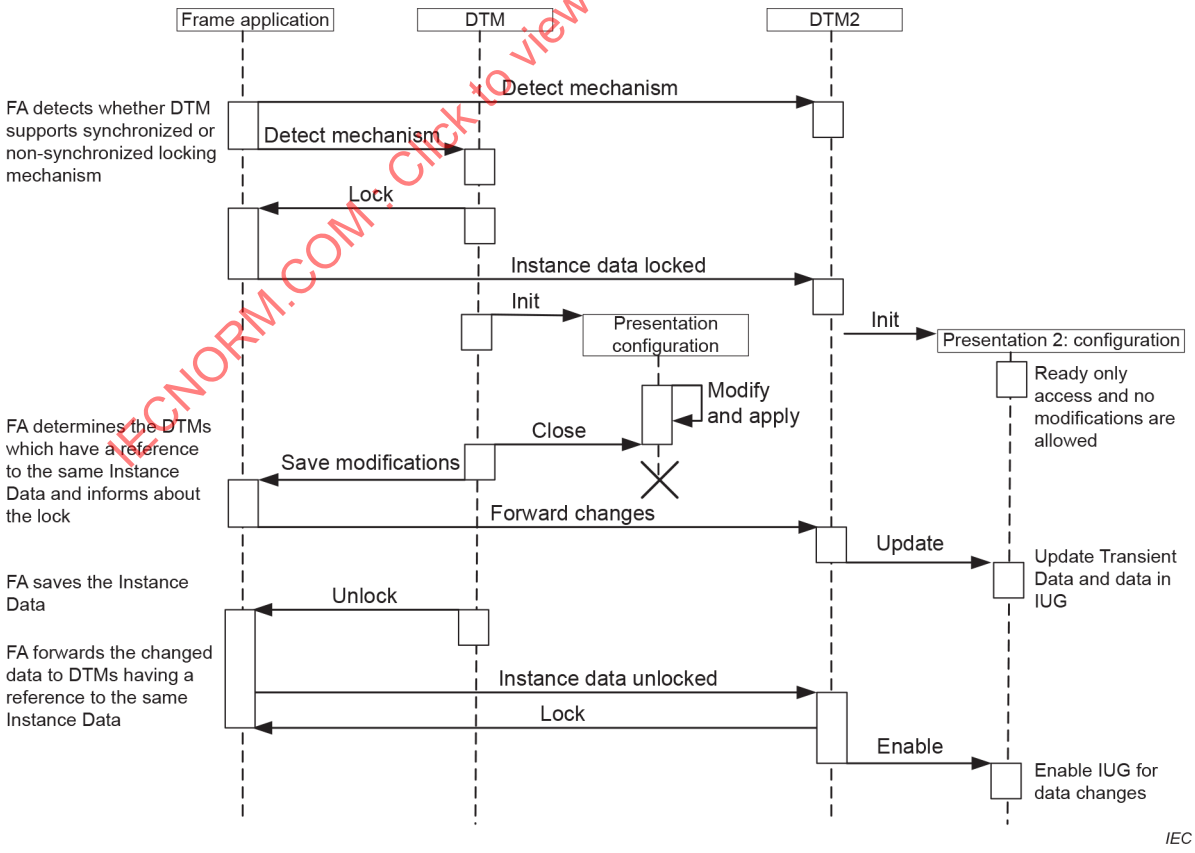


Figure 50 – Parameterization in the case of synchronized locking mechanism

8.5.3 Additional rules

Additional rules are:

- Before opening a Presentation object containing changeable parameters, the DTM shall try to lock the data set. If successful, all user input fields may be enabled. If unsuccessful, the user input fields shall be disabled. After closing all Presentation objects in the case of a locked data set, the DTM should ask to store the data set and unlock it after the Frame Application has stored the data set (after the Frame Application has called the storage services). The data set shall not be locked if the Presentation object does not contain changeable parameters, for example for applications such as observe.
- Before loading data from the device, the DTM shall also try to lock the dataset. After loading the data from the device, the DTM should request the FA to store its data and unlock data set after the FA has stored the data set. If the data set could not be locked, the DTM shall inform the user by an error message. In that case, the action will not be performed.
- In order to support multi-user environments, locks shall be kept as short as possible. This means, for example if a DTM locks its data set after instantiation and unlocks the data set only if it is terminated, this DTM is not suitable for use within a multi-user environment. A DTM shall lock its data set only if necessary and only during modification of the data. After the instance data is stored by the Frame Application and is not modified further, the DTM shall unlock its data set immediately. This enables concurrent access to the device data by other DTM instances within a multi-user environment.
- When the DTM receives a notification regarding an unlock of the data set, a Presentation object is active, which would allow to change the data set and if the access right of the current user allows changing the instance data set, the DTM shall ask the user if the user wants to have write access. When the user wants to have the write access, the DTM has to lock the data set. The DTM shall enable the input controls only if the DTM gets the lock.

8.6 Notification of changes

FDT defines a notification mechanism to inform other DTM instances for the same device about changes in the data set and to refresh those (synchronized DTMs). If the synchronization mechanism is not implemented (non-synchronized DTMs), changing the data set of a DTM on one PC can block other users on a different PC, because the DTM on the other PC needs to shutdown to reload the changed data. Shutting down the DTM on the other PC and loading the changed data set from the Project data base will take more time and will require more computing resources than an update by the notification mechanism.

Therefore, it is recommended that DTMs implement the notification mechanism for synchronized DTMs (see 8.5.2).

If a DTM has stored any changed data (after the Frame has called the storage service), the DTM shall call the service InstanceDataChanged. The DTM shall not call InstanceDataChanged before storing the instance data set.

If a DTM supports the notification mechanism for synchronized DTMs and when an event is received regarding changed parameters (service OnInstanceDataChanged), the DTM shall apply the new parameter values into the instance data set without requesting a lock. In this case, the storage state of the data set is persistent. Also all input controls in all opened graphical user interfaces shall be updated.

8.7 DTM instance data state machines

8.7.1 Instance data set overview

Subclause 8.7 describes the different states of DTM instance data with regard to modifications and persistence. The persistence mechanism itself is described in 4.9.

8.7.2 Modifications state machine

This state machine (see Figure 51) reflects the possible states of instance data concerning modifications.

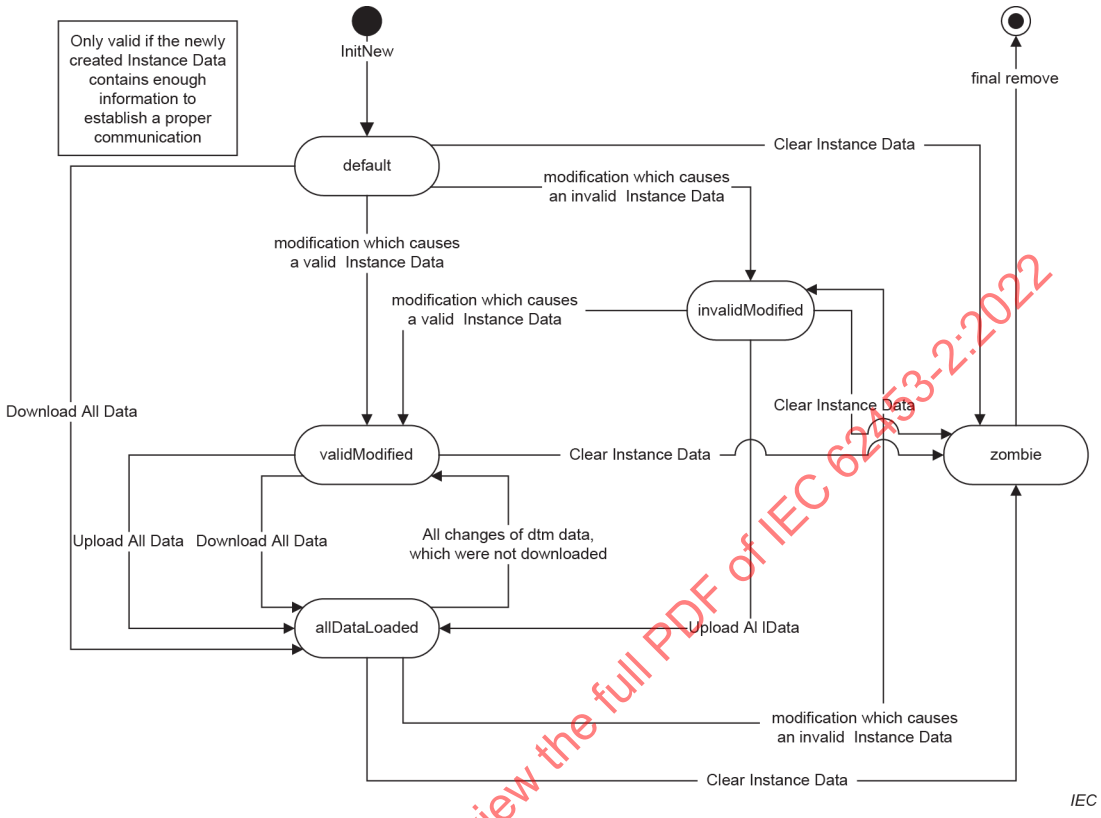


Figure 51 – Modifications state machine of instance data

The states of the modifications state machine are explained in Table 101.

Table 101 – Modifications state machine of instance data

State	Meaning
Default	The contents of the instance data are the default data. This state will typically appear after creation of new instance data
validModified	The instance data was modified in a consistent manner
invalidModified	The data set was modified. The data is not in a consistent state
allDataLoaded	The data was loaded from or into the related device. NOTE This state does not mean that the data within the device is equal to the data found within the related instance data. Due to the fact that a user can use tools out of the scope of FDT, FDT cannot guarantee such a state
zombie	The instance data is prepared to delete, no access to this data is allowed

The data set state can be used to verify which devices are in sync with their DTMs and which devices need to be reloaded. For example, 'allDataLoaded' indicates that a device is in sync with the DTM and shall not be loaded again. This state shall be changed only if data is really changed and not, for example if a lock is requested or a Presentation object is active without changing the data set, because reloading a device is a time consuming action.

8.7.3 Persistence state machine

This state machine reflects the possible states of the instance data concerning the persistence of the data (see Figure 52).

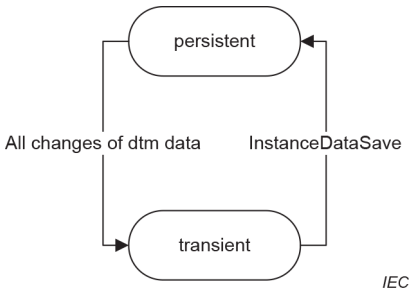


Figure 52 – Persistence state machine of instance data

The states of the persistence state machine are explained in Table 102.

Table 102 – Persistence state machine of instance data

State	Meaning
persistent	The content of the instance related data is persistent. This state will typically appear after the data was saved by the DTM using the persistence service SaveInstanceData (see 7.7.5.1) of the Frame Application
transient	The instance data was modified. These changes are not saved in a persistent way. All modified data within the DTM is temporary and not synchronized with other DTM instances or with the Frame Application

A DTM shall request to store the instance data set at a minimum if one of the following conditions occurs:

- if the last Presentation object, which is able to change parameter values has been terminated;
- after parameters have been read from or written to the device (services ReadDataFromDevice / WriteDataToDevice);
- after parameter values are changed by another component (InstanceDataWrite) and no Presentation object which is able to change parameter values is active.

8.7.4 Modification in device

8.7.4.1 General

It is useful to keep the device data set and the DTM instance data set in sync. When invoked in online mode, the DTM should ask the user whether the data in the DTM and in the device should be synchronized. This behavior is mandatory for Presentation objects with applicationId "fdtOfflineParameterize" and "fdtOnlineParameterize".

The mandatory flag <fdt:modifiedInDevice> will present the synchronization state.

8.7.4.2 Presentation fdtOnlineParameterize

After the user closes the DTM user interface, the DTM shall ask the user if the instance data set and the current device configuration should be synchronized. This may overwrite already existing parameter modifications within the instance data set (status not equal to 'allDataLoaded', refer to 8.7.3). If this is the case, the DTM shall include this information in the question to the user.

If the user confirms the synchronization, the DTM shall load the complete device data set into the instance data set. If the DTM can guarantee consistency of device data and instance data set, it could upload only the modified device parameters under consideration of the device-specific business rules.

8.7.4.3 Presentation fdtOfflineParameterize

If the user closes the DTM user interface and if the DTM can connect to the device (meaning the DTM is in a state of 'communication allowed'), the DTM shall ask the user if a complete write should be initiated to synchronize the instance data set and the current device configuration.

If the user confirms the download, the DTM shall download the complete instance data set into the device. If the DTM can guarantee consistency of the device data and instance data set, it could write only the modified parameters under consideration of the device-specific business rules.

8.7.5 Storage life cycle

Table 103 describes the states of the instance data set during a life cycle of a DTM. The states of the instance data set are defined by dataSetState (see Figure 51) and storageState (see Figure 52).

Table 103 – Example life cycle of a DTM

Scenario	storageState	dataSetState	modifiedInDevice	Remarks
DTM is initialized	transient	default	False	
After DTM is initialized and storing the data	persistent	default	False	
After failed read of device data	persistent	default	False	No change of the states, because due to the unsuccessful read the data set is not changed
After successful read of device data	transient	allDataLoaded	False	
After reading from device and storing the data set	persistent	allDataLoaded	False	
After modification of device data set via e.g. the GUI Online Parameterization	transient, because flag <modifiedInDevice > has changed	validModified	True	The device accepted new data, therefore a read from the device is necessary in order to synchronize the data sets
After storing the data set	persistent	validModified	True	
After read data from device	transient, because dataSetState is changed	allDataLoaded	False	
After modification of instance data set	transient	If changes comply to business rules of device: validModified Otherwise: invalidModified	False	
After storing the data set	persistent	validModified	False	
After release of the DTM	-/-	-/-		
After DTM started with persistent data	persistent	validModified	False	

8.8 Parent component handling redundant slave

A parent component, for example a Communication DTM, handling a redundant fieldbus is able to detect a Device DTM able to handle a redundant slave within the execution of the service ChildAdded by examining the information returned by the service NetworkManagementInfoRead of the Child DTM. The Communication DTM selects the redundant communication paths (either automatically or by means of a dialog) and sets the redundancy information to the Device DTM by calling NetworkManagementInfoWrite.

The Device DTM provides this redundancy information within the call to the service Connect to the Communication Channel. For each such opened connection the Communication Channel is able to use one of the available communication paths without need for further interaction with the Device DTM.

A Frame Application implementing the services related to redundancy (see 7.7.4) is able to manage topology information about redundant DTMs. In such a Frame Application, the topology view may show this redundancy information. On the other hand the Communication DTM and Device DTM of a redundant slave can be used in a Frame Application without knowledge about redundancy, because all redundancy functionality is handled by the Communication DTM and its Child DTMs.

The sequence related to the management of a redundant topology is shown in Figure 53.

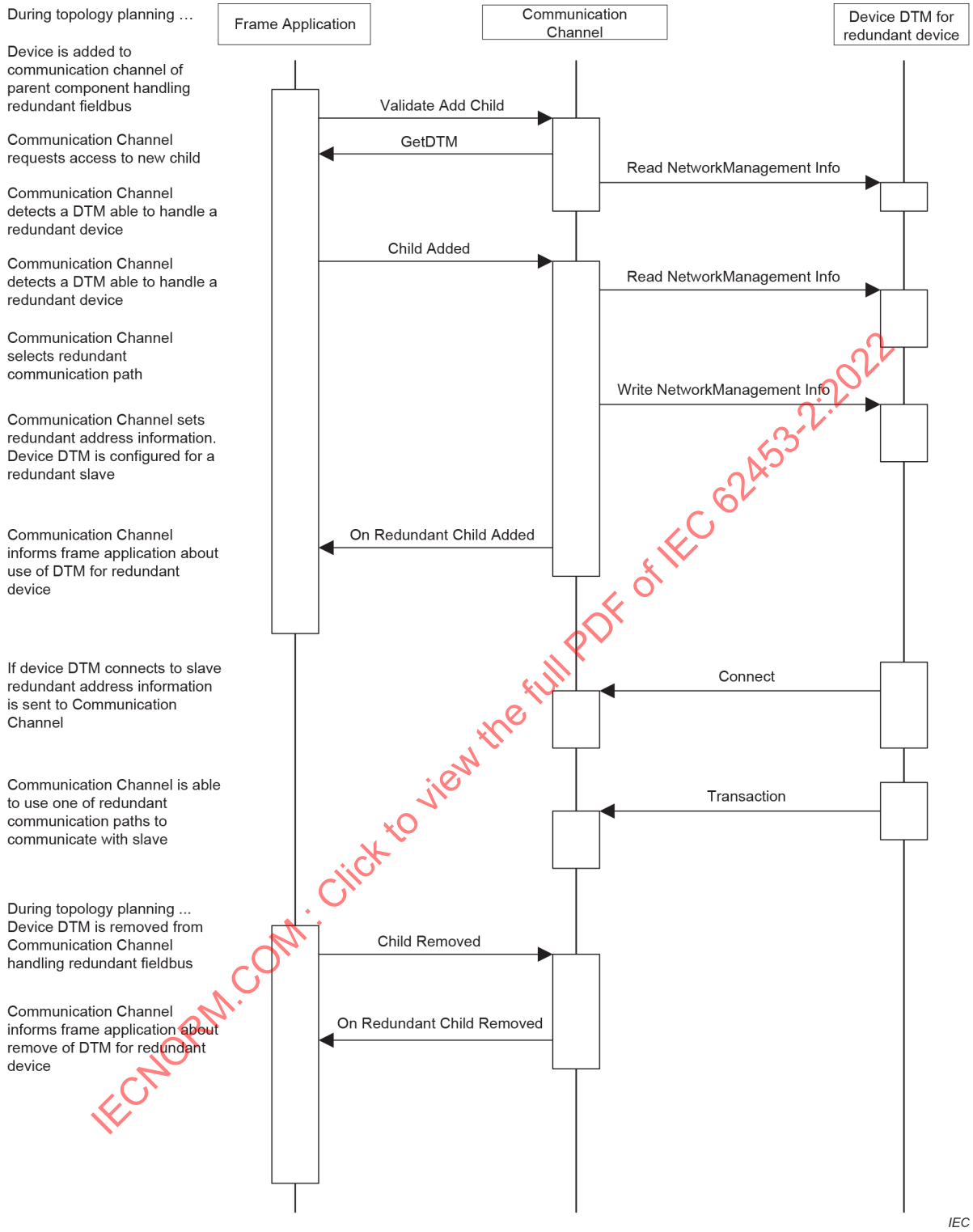


Figure 53 – Management of redundant topology

8.9 DTM upgrade

8.9.1 General rules

The first step to be resolved during the upgrade is to verify if the saved data set can be associated with the new DTM.

Each DTM should expose a unique identifier (dataSetId) specifying its data set format. A DTM can provide a list of compatible data set formats it can load. These dataSetIds are returned as part of the GetTypeInformation service.

If the ClassIdentifier of the DTM is not changed, it is up to the DTM to handle version upgrade. The list of supported data set formats may be ignored by the Frame Application when data is loading in this case. Additional check for data compatibility shall be done by the DTM when the data is loaded.

8.9.2 Saving data from a DTM to be upgraded

The first step is to store the information from the DTM that is to be upgraded.

The Frame Application needs to store additional information about the DTM:

- ClassIdentifier of a DTM;
- dataSetId of the stored data format;
- storage type;
- additional information about the device.

This information is associated with the stored data and can be used later by the Frame Application to identify and manage the data set.

The following diagram provides an illustration of that sequence (see Figure 54):

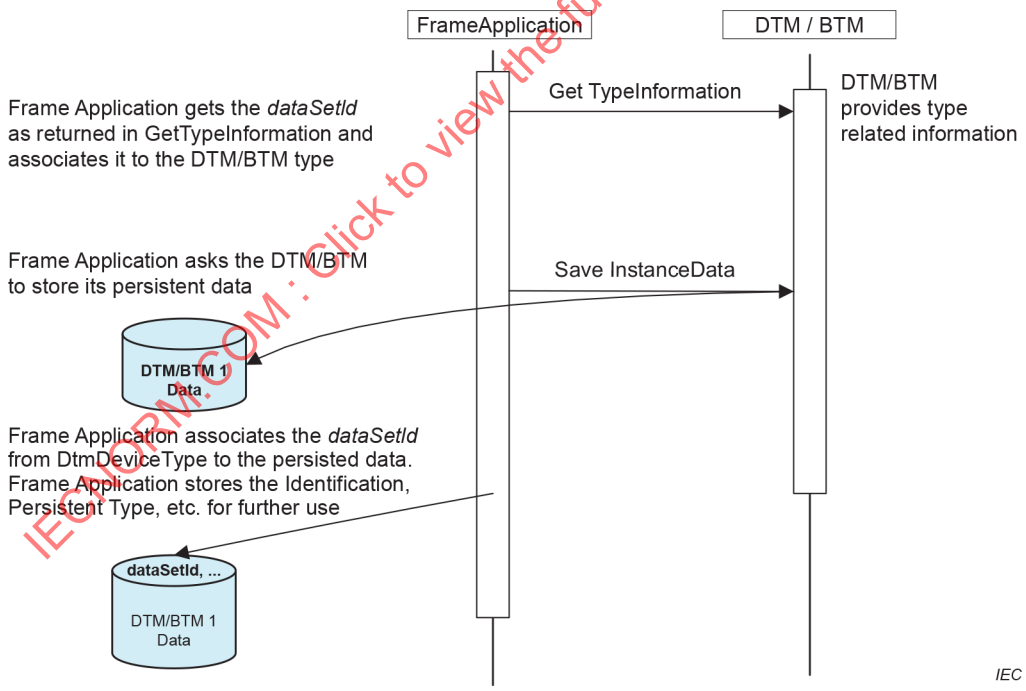


Figure 54 – Associating data to a dataSetId

8.9.3 Loading data in the replacement DTM

Since the stored data is associated to the storage information, it can be used later by the Frame Application to select a different DTM which is able to handle the stored data.

Figure 55 provides an illustration of that sequence.

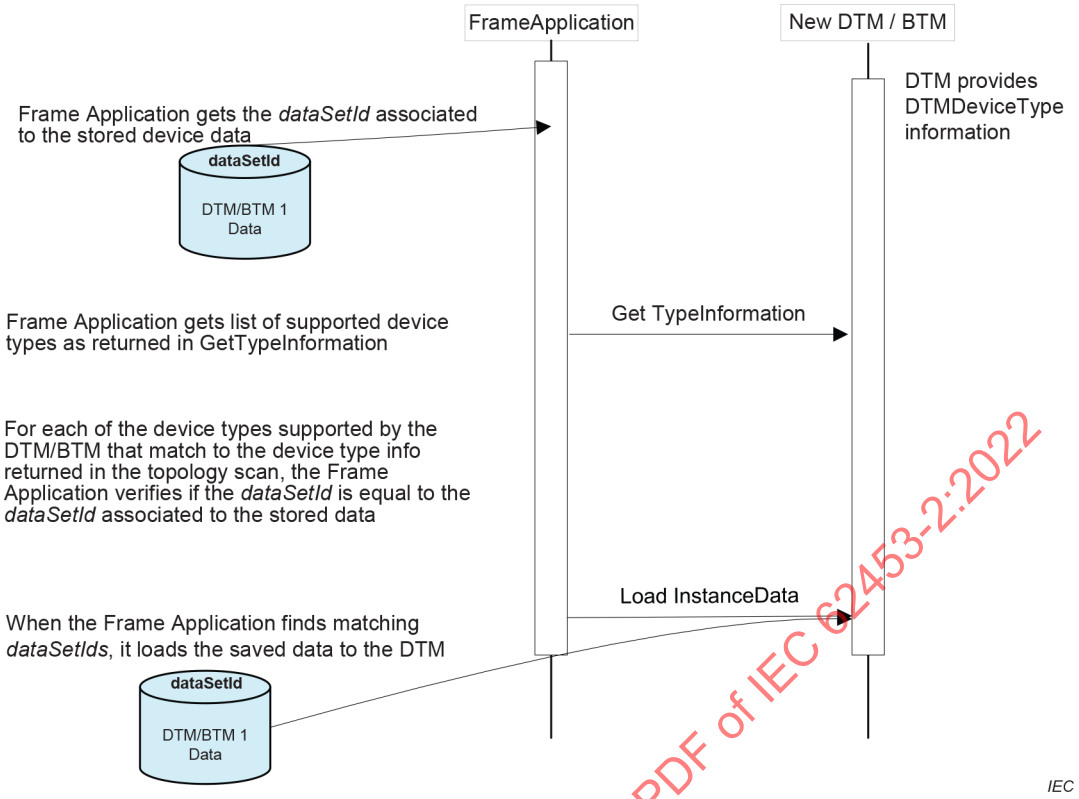


Figure 55 – Loading data for a supported dataSetId

Annex A
(normative)

FDT data types definition

A.1 General

This Annex A defines all the data types used for FDT. The data types are grouped according to functionality. Data types belonging together from a logical point of view are listed within one table. For each of these groups, a namespace is defined. The namespace is identified by a prefix. When a data type is referenced outside of the scope of its namespace, the prefix is added in front of the data type reference.

The names for data types can be composed from different words and are written in CamelCase, where an upper case letter indicates the beginning of a new word. Names of simple data types start with a lower case letter. Names of structured data types start with an upper case letter.

EXAMPLES

Simple data type 'address' defined in namespace 'fdt':	fdt:address
Simple data type 'semanticId' defined in namespace 'fdt':	fdt:semanticId
Structured data type "SemanticInformation" defined in namespace 'fdt':	fdt:SemanticInformation

For each data type the basic data type is defined as well as a description.

The name and the semantics of the data types as defined in the columns "Data type" and "Description" in the data type in Table A.1 to Table A.30 are normative. The column "Definition" provided in the data type tables contains informative content.

For structured data types, the column "Definition" consists of "Elementary data types", "Usage" and "Multiplicity". "Elementary data types" describes the structure of the data type and the data type of the sub-elements.

"Usage" describes how a sub-element is used within the structured data type.

Possible values for usage are:

O – optional (this data element shall not be provided, but may be provided);

M – mandatory (this data element shall be provided);

C – conditional (it depends on a condition, whether this data element is provided, the condition shall be described in the right column);

S – selective (this element and another element exclude each other; this may occur with a "choice of" structure or otherwise. If there is a condition other than "choice of", the condition shall be described in the Description column).

"Multiplicity" defines how often a sub-element may occur within the structured data type. The used syntax is defined by UML notation. Multiplicity is expressed in terms of lower-bound and upper-bound. The bounds are presented by integer numbers or by an asterisk ("*"). The meaning of an asterisk is 'unlimited'.

Most of the time, the information for Multiplicity may co-relate with the Usage information (e.g. the optional element may have a Multiplicity of [0..1], the mandatory element may have a multiplicity of [1..1]). However, there are cases where Multiplicity may provide additional information (e.g. in cases of conditional and selective usage).

A.2 Basic data types

The data type definition in the IEC 62453 series is based on data types defined in the IEC 61131 series. Table A.1 defines additional basic data types.

Table A.1 – Basic data types

Data type	Description
ClassIdentifier	Technology-specific identifier for a specific implementation class
LanguageID	Unique identifier for user interface localization. The ID is a standard international numeric abbreviation (e.g. German – Standard: 0x0407, English – United States: 0x0409)
ObjectReference	Technology-specific reference to an object instance
UUID	A worldwide unique ID
URI	A compact string of characters used to identify or name a resource. The terms URL and URN are context-dependent aspects of URIs, and rarely need to be distinguished. Examples for URI are: the URI syntax is essentially a URI scheme name such as "http", "ftp", "mailto", "urn", "file", etc., followed by a colon character, and then a scheme-specific part

A.3 General data types

Namespace: fdt

The simple data types (see Table A.2) and structured data types (see Table A.4) defined in Clause A.3 are used as a base for the definition of service-specific data types or as service arguments.

Table A.2 – Simple general data types

Data type	Definition	Description
address	STRING	Parameter Addressing information. The format of the value for this parameter is described for each communication protocol in the corresponding section of the specification. The protocol portion of the specification provides the guidelines for address attribute. Most of the protocols specify that the DTM shall expose the addressing information to the Frame Application
alarmType	enumeration (highHighAlarm | highAlarm | lowLowAlarm | lowAlarm)	Identifier for the alarm type to show the association between high alarm and low alarm and high-high alarm and low-low alarm
applicationDomain	STRING	The application domain that applies to the provided semanticId. This may be a protocol-specific ID or a different FDT-defined application domain
applicationId	enumeration (fdtAdjustSetValue | fdtAssetManagement | fdtAuditTrail | fdtConfiguration | fdtDiagnosis | fdtForce | fdtManagement | fdtObserve | fdtOfflineCompare | fdtOfflineParameterize | fdtOnlineCompare | fdtOnlineParameterize | fdtIdentify | fdtCalibration | fdtMainOperation | fdtNetworkManagement)	Identification of the current application context
binData	ARRAY OF USINT	Variable containing binary data
bitLength	UDINT	Length of a binary variable as bit count
bitPosition	UDINT	Position of a bit within an enumeration (0 based position)

Data type	Definition	Description
boolValue	BOOL	Variable for configured static boolean data such as alarm value
build	INT	Build part of the FDT-Specification version on which the implementation of a DTM is based. For example, for FDT-Specification 1.2.1.0 it shall be set to '0'
busAddress	STRING	
busRedundancy	UDINT	Number of redundant fieldbus interfaces
byteArray	ARRAY OF USINT	Data type used to transfer binary data
byteLength	UDINT	Number of bytes to describe data types like string
channelId	STRING	Id of a channel
channelMode	enumeration (communication | moduleSlot | processValue)	Type information enumeration for a channel
channelPath	STRING	Returns the path that identifies the channel within the device. The string shall not be empty. It always starts with the systemTag of the device instance followed by the channel id. The DTM has to guarantee that the path is unique for a device instance. The channel path is the base information to handle the system structure. <systemTag>/<id> Furthermore, the channelPath contains the information about where to find the channel within the information available via [GetParameters] and so, at least in the case of a monolithic DTM, the information whether the channel belongs to a device or module. In the case of Communication Channels there are some special rules for building the channelPath. How to generate the channelPath of a Communication Channel, which also acts as a Process Channel for data that it receives from a Process Channel of a child device, depends on the functionality of the channel. If the channel is passive, meaning that it receives its data from a child device and provides this data without any changes within its own [ReadChannelData] information, then the channelPath shall be built out of the system tag and channel id of its own device instance and the system tag of the child device and the id of the marshalled channel. <systemTag>/<id>//<systemTag>/<id> If the channel is active, meaning that it receives its data from a child device for processing and provides the result within its own [ReadChannelData] information, then the channelPath shall be built out of the system tag and channel id of the own device instance. <systemTag>/<id> This allows navigation through the internal channel assignment of a device with gateway functionality
classificationDomainId	enumeration (powerDistribution | motionControl | measurement | operatorInterface | modulesAndControllers)	Device classification domain group according to IEC TR 62390:2005, Annex G. See Table A.3 below

Data type	Definition	Description
classificationId	enumeration (flow | level | pressure | temperature | valve | positioner | actuator | nc_rc | encoder | speedDrive | hmi | analogInput | analogOutput | digitalInput | digitalOutput | electrochemicalAnalyser | dtmSpecific | universal | analyser | remoteIO | analogCombinedIO | digitalCombinedIO | recorder | controller | angle | limitSwitch | converter | motor | | switchboard | circuitBreaker | powerMonitoring | distributionPanel | contactor | protectionStarter | softStarter | drive | axisControl | motorControlCenter | controlValve | electrical | density | quality | speedOrRotaryFrequency | radiation | weightMass | distanceOrPositionPresence | pushButton | joystick | keypad | pilotLight | stackLight | display | combinedButtonsAndLights | operatorStation | generalInput | generalOutput | combinedInputOutput | relay | timer | scanner | programmableController)	Unique identifier for classification of a channel or device according to its primary function. 'classificationId' should be selected from the table 'Device classification' according to IEC TR 62390:2005, Annex G' and use the corresponding 'classificationDomainId' attribute for it. If a suitable classification ID does not exist in that table, it should be selected from the 'FDT classification ID table' and use it without the 'classificationDomainId' attribute. See Table A.3 below
communicationError	enumeration (abort | busy | invalidCommunicationReference | noConnection | noParallelServices | noPendingRequest | unknownError | timeout | dtmSpecific | notSupportedFeature | sequenceTimeExpired)	Fieldbus protocol independent error occurred during communication. This kind of error is used especially if an error occurred during nested communication. The fieldbus-specific communication error is part of the fieldbus-specific parts of this specification. Communication errors detected by a communication component, e.g. a Communication DTM, shall be handled within the data returned to the child component. All errors returned by a device as a result of a fieldbus transaction should be returned by protocol-specific read response or write response elements, typically by using fieldbus-specific status and errors. All errors detected by the Communication DTM shall be mapped to one of the fdt:communicationError enumeration values (e.g. abort busy invalidCommunicationReference noConnection noParallelServices noPendingRequest unknownError timeout dtmSpecific notSupportedFeature) and returned as an fdt:CommunciationError element to the Child DTM. A DTM shall be able to handle both types of error responses for a transaction request sent to the parent component
communicationReferenceID	UUID	Identifier for a communication link to a device. This identifier is allocated by the communication component during the connect. The communication reference has to be used for all the following communication calls
communicationType	enumeration (supported | required)	Indicates the type of the protocol support: – 'supported': bus protocol which can be provided by the DTM at a channel – 'required': bus protocol which will be expected by the DTM from the Parent DTM NOTE The actual supported bus protocols depend on the configuration of the DTM. This data type shows the possible supported, not the actual available protocols
dataSetID	UUID	Unique identifier of the data set version

Data type	Definition	Description
dataSetState	enumeration (default | validModified | invalidModified | allDataLoaded)	State of an instance data set concerning modifications (refer to 4.9)
dataType	enumeration (byte | float | double | int | unsigned | enumerator | bitEnumerator | index | ascii | packedAscii | password | bitString | hexString | date | time | dateAndTime | duration | binary | structured | dtmSpecific)	Identifier for the data type of a transferred variable
dataTypeDescriptor	STRING	Description of data type
date	DATE	Date variable within an element
description	STRING	A human readable description of the supported and current data set format. Can be used to provide additional information to the user in case of partial level of support
descriptor	STRING	Human readable description within the context of an element
deviceGraphicState	enumeration (device | diagnosis | oem)	List of possible device states used in the DeviceIcon and DeviceBitmap element. Possible states are: – device: symbolic representation of the device – diagnostic: symbolic representation of device if there is online diagnosis available – oem: symbol representation of device in special operating modes (e.g. OEM service) The Frame Application should display the correct picture according to protocol-specific rules
deviceTypeId	UDINT	Unique identifier for a device type within the name space of the fieldbus-specification
deviceTypeInformation	STRING	Additional device type information supplied with a device. For example, a PROFIBUS[2] device has to provide its GSD information as human readable string at this attribute. See protocol specific definitions in the relevant IEC 62453-3xy document. The information shall be based on the current locale according to the usage of DTM service SetLanguage NOTE The GSD information is accessible via the services InstanceDataInformation and GetTypeInformation
deviceTypeInformationPath	URI	Path to the file containing the information which is provided via the attribute 'deviceTypeInformation'. It is recommended to use this attribute for providing additional protocol-specific descriptions of the device type. The use of this attribute is specified in the protocol-specific documents (IEC 62453-3xy)
deviceTypeVariant	STRING	Manufacturer-specific device type variant definition string
deviceTypeVariantInfo	STRING	Contains additional information for manufacturer-specific device type variant definition
display	STRING	Carries a human readable display string for tasks like documentation
displayFormat	STRING	Describes the display format for a display attribute (e.g. "%3,2f " for a float value)

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

Data type	Definition	Description
documentClassification	enumeration (help | technicalDocumentation | orderingInformation | miscellaneous)	Specifies the subject of the document
documentLanguageId	UDINT	Identifier for the language of the document. The language id is defined as a Windows locale identifier (LCID)
dtmDevTypeID	UUID	dtmDevTypeID is a DTM-specific unique identifier of the software related to the device type. For the same device type, the value remains unchanged although some identification information in the DtmDeviceType element is updated. This can be a result of DTM update. The same dtmDevTypeID value shall always be related to the same device as in the previous DTM version (e.g. to provide backward compatibility)
dtmDevTypeIDVersion	UDINT	Version number of the dtmDevTypeID. It is also used to indicate that the information related to DtmDeviceType is changed. Frame Application can use this information, e.g. to update the DTM related information in the DTM library. An example: When a DTM is DD based, an upgrade of DD will cause the change of dtmDevTypeIDVersion without changing the dtmDevTypeID
errorCode	ARRAY OF USINT	Status information according to fieldbus specification
errorMessage	STRING	Human readable error message
filePath	URI	Path to a file (e.g. document, executable)
frameApplicationTag	STRING	Frame Application-specific tag used for identification and navigation. The DTM should display this tag at channel-specific user interfaces
functionId	DINT	Identifier of a function provided by a DTM
help	STRING	Human readable help string for a document
id	STRING	Unique identifier for an element. This identifier is used within collections for the direct access of elements. This id shall be unique within the namespace of a device instance
idref	STRING	Reference to an element specified by the attribute 'id'. The identifier is used within collections for the direct access of elements
index	UDINT	Index within an enumerator
invokeId	UUID	Identifier of a context of a service call, if the service provider handles several contexts in parallel. The identifier is used to reference former context calls
label	STRING	Human readable label for a document
languageId	UDINT	Identifier for a supported language or the language with which a DTM should interact. See also DTM service SetLanguage
levelOfSupport	enumeration (full | partial)	Provides an indication about the level of support of the supported data set version: – full – the data set is fully supported – partial – some of the information in the data set cannot be used in the upgrade procedure. Additional information should be provided in the description attribute. Can be used to warn the user that some values can be lost
major	INT	Major part of the FDT specification version on which the implementation of a DTM is based. For example, for FDT specification 1.2.1.0 it shall be set to '1'

Data type	Definition	Description
manufacturerId	UDINT	Unique identifier for a manufacturer within the name space of the fieldbus specification
minor	INT	Minor part of the FDT specification version on which the implementation of a DTM is based. For example, for FDT specification 1.2.1.0 it shall be set to '2'
modifiedInDevice	BOOL	Flag to provide more information about the current state of the instance data set. TRUE indicates that parameters have been changed in the device but not in the instance data set (e.g.: see use case Online parameterization, DTM services to access the device data). 'modifiedInDevice' flag shall be set only once if the data in the device has been changed. In the case of successful Upload or Download of the complete data set, the 'modifiedInDevice' flag shall be reset (set to FALSE). Data in the device may also be modified directly by a tool out of the scope of FDT. In this case, the flag 'modifiedInDevice' should not be set. Data set shall be locked before an application is started, which may need to change the flag 'modifiedInDevice' (the flag 'modifiedInDevice' is part of the instance data set and cannot be changed if the data set is not locked)
name	STRING	Human readable name within the context of an element
number	INT	Number variable such as float, integer or other numeric data types
onlineStatus	enumeration (communicationSet | goingOnline | goingOffline | online)	Information about online state of a DTM describing the 'communication allowed' substates
operationPhase	enumeration (notSupported | engineering | commissioning | runtime | service)	Information about the current operation phase
orderCode	STRING	Order code of the device variant
parameters	STRING	Contains the parameters to be passed to the application if the file attribute specifies an executable file
path	URI	Path to the icon for a device
percent	USINT	State of progress 0 %..100 %
physicalLayer	UUID	CATID for description of a physical layer of a fieldbus
physicalLayerName	STRING	Human readable description for the physical layer
protectedByChannelAssignment	BOOL	Defines if the channel is set to read only by the Frame Application. This is set to TRUE if a channel assignment exists
protocolId	UUID	Identification of a protocol. Standard values for this data type are defined in the protocol-specific documents (IEC 62453-3xy)
protocolIdName	STRING	Human readable name of a protocol, associated with a protocolId. Standard values for this data type are defined in the protocol-specific documents (IEC 62453-3xy)
readAccess	BOOL	Specifies whether the value can be read from a device
reference	STRING	Reference to a variable of a structure
release	INT	Release part of the FDT specification version on which the implementation of a DTM is based. For FDT specification 1.2.1.0 it shall be set to '1'

Data type	Definition	Description
semanticId	STRING	Semantic identifier for an element. This identifier shall be unique at least within the context of an element. By using this attribute, e.g. a Frame Application is able to get the information regarding the meaning and usage of a single data structure. The definition regarding the identifier is protocol-specific and described in protocol-specific annexes. Furthermore, the following protocol-independent semanticIds are defined to cover the network information: – fdt::DeviceAddress – fdt:busMasterConfigurationPart A parameter with one of these semanticIds shall be the same parameter as the corresponding information (e.g. bus address) provided by the service NetworkManagementInfoRead (see 7.2.11.1)
serviceSucceeded	BOOL	TRUE indicates succeeded service, FALSE indicates failed service
showProgress	BOOL	Set to TRUE, if the progress should be displayed, otherwise the progress would not be shown and an open progress bar shall be closed within the Frame Application
signalType	enumeration (input | output)	Specifies a signal as input or output
staticValue	INT	Variable for configured static data such as an alarm value
statusFlag	enumeration (ok | warning | error | invalid)	Identifier for the current status of a device or module
storageState	enumeration (persistent | transient)	State of an instance data set concerning the persistent state (refer to 4.9)
string	STRING	String variable within an element
subDeviceType	STRING	Manufacturer-specific unique identifier for a device type within the name space of the device type id. This parameter shall be passed to the DTM during initialization via the Initialize service to advise a pre-configuration for the requested subtype. For example, the same transmitter can be pre-configured for level or flow measuring
substituteType	enumeration (lastValue | lastValidValue | upperRange | lowerRange)	Type of an substitute value
systemTag	STRING	Unique identification of the DTM within the Frame Application
tag	STRING	Unique identifier for a device, module, or channel
time	DATE_AND_TIME	Time variable within an element
url	URI	Contains a URL to a document in the Web
vendor	STRING	Human readable description of the vendor of the component
version	STRING	Human readable description of the version of component the such as "1.0"
windowTitle	STRING	Window title required by the Frame Application
writeAccess	BOOL	Specifies whether the value can be written into a device

Table A.3 defines the value range for the classification of the device types.

Table A.3 – Definition of classificationId enumeration values

Domain ()		Sub-domain
classificationDomainId	**IEC domain group name**	**classificationId**
General FDT	<none>	valve
		actuator
		nc_rc
		encoder
		speedDrive
		hmi
		analogInput
		analogOuput
		digitalInput
		digitalOutput
(..continuing General FDT)		electrochemicalAnalyser
		dtmSpecific
		universal
		analyser
		remoteIO
		analogCombinedIO
		digitalCombinedIO
		recorder
		controller
		angle
		limitSwitch
		converter
		motor
powerDistribution	Power distribution	switchboard
		circuitBreaker
		powerMonitoring
		distributionPanel
motionControl	Motion control	contactor
		protectionStarter
		softStarter
		drive
		axisControl
		motorControlCenter
		motorMonitoring
		positioner
		controlValve

Domain ()		Sub-domain
classificationDomainId	**IEC domain group name**	**classificationId**
measurement	Detection, measurement	electrical
		density
		flow
		level
		quality
		pressure
		speedOrRotaryFrequency
		radiation
		temperature
		weightMass
		distanceOrPositionOrPresence
operatorInterface	Dialogue / operator interfaces	pushButton
		joystick
		keypad
		pilotLight
		stackLight
		display
		combinedButtonsAndLights
		operatorStation
modulesAndControllers	Logic / universal I/O modules and controllers	generalInput
		generalOuput
		combinedInputOuput
		relay
		timer
		scanner
		programmableController

Table A.4 – General structured data types

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
Alarm	STRUCT			Description of an alarm
	alarmType	M	[1..1]	
	Unit	O	[0..1]	
	choice of	O	[0..1]	
	StaticValue	S	[1..1]	
	NumberData	S	[1..1]	
	AlarmEnumerationEntries	S	[1..1]	
	ChannelReferences	S	[1..1]	
AlarmEnumerationEntries	STRUCT			Collection of alarm enumeration entries
	AlarmEnumerationEntry	M	[1..*]	
AlarmEnumerationEntry	STRUCT			Alarm enumeration entry
	bitPosition	M	[1..1]	
	name	M	[1..1]	
	boolValue	M	[1..1]	
	descriptor	O	[0..1]	
Alarms	STRUCT			Collection of alarms
	Alarm	M	[1..*]	
ApplicationId	STRUCT			FDT global application id coded as an enumeration
	applicationId	M	[1..1]	
FDTApplicationIds	STRUCT			Collection of application id
	ApplicationId	O	[0..*]	
BinaryVariable	STRUCT			Element containing binary data
	binData	M	[1..1]	
BitEnumeratorEntries	STRUCT			Collection of EnumerationEntry
	EnumeratorEntry	M	[1..*]	
BitEnumeratorVariable	STRUCT			Current EnumeratorEntry and the collection of possible EnumeratorEntries
	BitVariable	M	[1..1]	
	BitEnumeratorEntries	O	[0..1]	
BitVariable	STRUCT			Selected element of an enumeration
	EnumeratorEntry	O	[0..*]	
BoolValue	STRUCT			Variable for configured static boolean data
	boolValue	O	[0..1]	
BusCategories	STRUCT			Collection of BusCategory
	BusCategory	M	[1..*]	
BusCategory	STRUCT			Description of a Fieldbus
	protocolId	M	[1..1]	
	protocolIdName	O	[0..1]	
	CommunicationTypeEntry	O	[0..*]	
	PhysicalLayer	O	[0..*]	
BusRedundancy	STRUCT			Number of redundant fieldbus interfaces
	busRedundancy	M	[1..1]	

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
ChannelInformation	STRUCT			Type information for an FDT channel. This element is used within a document returned by InstanceDataInformation service
	BusCategories	M	[1..1]	The BusCategories element should contain the list of supported fieldbus protocols of this channel
	ChannelModes	M	[1..1]	
ChannelMode	STRUCT			Type information element for a channel
	channelMode	M	[1..1]	
ChannelModes	STRUCT			List of ChannelMode elements
	ChannelMode	M	[1..*]	
ChannelObjectReference	ObjectReference			Reference to a Channel object
ChannelReference	STRUCT			Reference to an object identified by its id
	idref	M	[1..1]	
	ChannelInformation	O	[0..1]	
ChannelReferences	STRUCT			Collection of ChannelObjectReferences
	ChannelReference	M	[1..*]	
ChannelType	STRUCT			Description of the channel component
	VersionInformation	M	[1..1]	
	BusCategory	O	[0..1]	
ClassificationId	STRUCT			Classification id. See tables below
	classificationId	M	[1..1]	
ClassificationIds	STRUCT			Collection of classificationId elements. See tables below
	classificationDomainId	O	[0..1]	
	ClassificationId	M	[1..*]	
CommunicationClientObjectReference	ObjectReference			Reference to a communication client object
CommunicationData	STRUCT			Variable used to transfer binary communication data
	byteArray	M	[1..1]	
CommunicationError	STRUCT			Description of a fieldbus protocol independent error that occurred during nested communication with:
	communicationError	M	[1..1]	Protocol independent error
	tag	M	[1..1]	The tag of the Communication Channel's device
	errorCode	O	[0..1]	Fieldbus-specific error code
	descriptor	O	[0..1]	Error description
CommunicationObjectReference	Technology-specific ObjectReference			Reference to communication object. This data type depends on the used implementation technology (see for example IEC TR 62453-41, IEC TR 62453-42, or IEC TR 62453-43[3])
CommunicationTypeEntry	STRUCT			Enumeration element for the communication type
	communicationType	M	[1..1]	
CurrentDatasetFormat	STRUCT			A current version of the data set used for saving the data
	dataSetID	M	[1..1]	

[3] Under consideration.

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
	description	O	[0..1]	
DataSetFormats	STRUCT			Data formats of the persistent data used and supported by the DTM
	CurrentDatasetFormat	M	[1..1]	
	SupportedDatasetFormat	O	[0..*]	
Deadband	STRUCT			Deadband is the amount of value changes that triggers for example new trend values
	number	M	[1..1]	
DeviceBitmap	STRUCT			Bitmap for a device in BMP format (70 × 40 pixels (width × height), 16 colors)
	deviceGraphicState	M	[1..1]	
	path	M	[1..1]	
DeviceIcon	STRUCT			Icon for a device
	deviceGraphicState	O	[0..1]	
	path	M	[1..1]	
DeviceTypeVariant	STRUCT			Contains manufacturer-specific device type variant information. A device type variant is a property of the physical device that has no influence on the software (i.e. flange material, Ex certificate, etc.). However, some Frame Applications will use this information for documentation purposes
	deviceTypeVariant	M	[1..1]	
	deviceTypeVariantInfo	O	[0..1]	
	orderCode	O	[0..1]	
DeviceTypeVariants	STRUCT			Collection of DeviceVariant elements. The DeviceVariants element should only be used within the context of the DTM information data
	DeviceTypeVariant	M	[1..*]	
Display	STRUCT			Display variable
	string	M	[1..1]	
DisplayContext	Technology-specific display context			Contex information for display. This data type depends on the used implementation technology (see for example IEC TR 62453-41, IEC TR 62453-42, or IEC TR 62453-43)
DocumentExe	STRUCT			Defines an executable, which is used to show DTM documents
	file	M	[1..1]	
	parameters	O	[0..1]	
DocumentFile	STRUCT			Defines a file document
	filePath	M	[1..1]	
DocumentUrl	STRUCT			Defines a URL to a document in the Web Implementation hint: The file, URL and executable documents can be implemented by using the ShellExecute() function
	url	M	[1..1]	

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
DtmDeviceType	STRUCT			Description of a device type
	readAccess	O	[0..1]	
	writeAccess	O	[0..1]	
	manufacturerId	O	[0..1]	
	deviceTypeId	O	[0..1]	
	subDeviceType	O	[0..1]	
	deviceTypeInformation	O	[0..1]	
	deviceTypeInformationPath	O	[0..1]	
	classificationId	O	[0..1]	
	dtmDevTypeID	M	[1..1]	
	dtmDevTypeIDVersion	M	[1..1]	
	VersionInformation	M	[1..1]	
	SupportedLanguages	M	[1..1]	
	BusCategories	O	[0..1]	
	DeviceIcon	O	[0..*]	
	DtmDocuments	O	[0..1]	
	DeviceBitmap	O	[0..*]	
	ClassificationIds	O	[0..1]	
	DataSetFormats	O	[0..1]	
	choice of	O	[0..1]	
	DeviceTypeVariants	S	[1..1]	
	DeviceTypeVariant	S	[1..1]	
DtmDeviceTypes	STRUCT			Collection of device types
	DtmDeviceType	M	[1..*]	
DtmDocument	STRUCT			Definition of a document, which is provided by a DTM and displayed by the Frame Application
	label	M	[1..1]	
	help	O	[0..1]	
	documentClassification	M	[1..1]	
	documentLanguageId	O	[0..1]	
	choice of	M	[1..1]	
	DocumentFile	S	[0..1]	
	DocumentUrl	S	[0..1]	
	DocumentExe	S	[0..1]	
DtmDocuments	STRUCT			List of DTM documents. This tag allows a DTM to publish documents
	collection of	M	[1..1]	
	DtmDocument		[1..*]	
DtmObjectReference	ObjectReference			Reference to a DTM
DtmVariable	STRUCT			Variable description with name, value, range, etc.
	SemanticInformation	M	[1..*]	The DTM shall provide a SemanticInformation element for all supported fieldbus protocols of the DTM instance
	name	M	[1..1]	
	descriptor	O	[0..1]	
	Value	M	[1..1]	
	Unit	O	[0..1]	
	Ranges	O	[0..1]	
	statusFlag	O	[0..1]	
	StatusInformation	O	[0..1]	

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
DtmVariableReference	STRUCT			Reference to a DTM variable
	reference	M	[1..1]	
DtmVariables	STRUCT			Collection of DTM variables
	SemanticInformation	O	[0..*]	
	name	M	[1..1]	
	descriptor	O	[0..1]	
	collection of	M	[1..1]	
	DtmVariables		[0..*]	
	DtmVariable		[0..*]	
EnumeratorEntries	STRUCT			Enumeration element
	EnumeratorEntry	M	[1..*]	
EnumeratorEntry	STRUCT			Element of an enumeration
	index	M	[1..1]	
	name	M	[1..1]	
	descriptor	O	[0..1]	
EnumeratorVariable	STRUCT			Current EnumeratorEntry and the collection of possible EnumeratorEntries
	Variable	M	[1..1]	
	EnumeratorEntries	O	[0..1]	
FDTVersion	STRUCT			Definition of the element which specifies the version information on which the implementation of a DTM is based
	major	M	[1..1]	
	minor	M	[1..1]	
	release	M	[1..1]	
	build	M	[1..1]	
FrameObjectReference	ObjectReference			Reference to the Frame Application
FrameVersion	STRUCT			Describes the version of the Frame Application
	FDTVersion	M	[1..1]	
LanguageId	STRUCT			Contains the languageId (refer to languageId)
	languageId	M	[1..1]	
LowerRange	STRUCT			Definition of a lower range element
	choice of	O	[0..1]	
	NumberData	S	[1..1]	
	StringData	S	[1..1]	
	TimeData	S	[1..1]	
	ChannelReference	S	[1..1]	
LowerRawValue	STRUCT			A numeric entry which reflects the actual value transferred via a fieldbus. The value is mapped to the related range (LowerRangeValue). For example, if a PROFIBUS device maps a range from 10 mbar to 100 mbar to an integer of value 1 024 to 4 096 the 'LowerRawValue' contains 1 024, the 'UpperRawValue' contains 4 096
	number	M	[1..1]	
NumberData	STRUCT			Number variable such as float, integer or other numeric data types
	number	M	[1..1]	
PhysicalLayer	STRUCT			Unique identifier for a physical layer of a fieldbus
	physicalLayer	M	[1..1]	
	physicalLayerName	O	[0..1]	

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
PresentationObject-Reference	ObjectReference			Reference to a Presentation object
ProgressInformation	STRUCT			Progress information with:
	description			Description of the running process
	percent			
	showProgress			
Range	STRUCT			Describes the valid range of a process variable
	LowerRange	M	[1..1]	
	UpperRange	M	[1..1]	
	Unit	O	[0..1]	
	LowerRawValue	O	[0..1]	
	UpperRawValue	O	[0..1]	
Ranges	STRUCT			Collection of ranges
	readAccess	O	[0..1]	
	writeAccess	O	[0..1]	
	Range	M	[1..*]	
SemanticInformation	STRUCT			The element provides semantic information for a data object. For each object at least one SemanticInformation element with an FDT-defined protocol-specific semanticId shall be provided. The DTM shall provide a SemanticInformation element for all supported fieldbus protocol of the DTM instance
	applicationDomain	M	[1..1]	
	semanticId	M	[1..1]	
	address	O	[0..1]	
StaticValue	STRUCT			Variable for configured static data such as an alarm value
	staticValue	M	[1..1]	
StatusInformation	STRUCT			Current status information of a device or module
	readAccess	O	[0..1]	
	writeAccess	O	[0..1]	
	EnumeratorEntry	M	[1..*]	
StringData	STRUCT			String variable
	string	M	[1..1]	
StorageObject	ObjectReference			Technology-specific object used in data save and load services
StructuredElement	STRUCT			Variable as display value or as reference to a variable with data length information
	bitLength	M	[1..1]	
	choice of	M	[1..1]	
	Display	S	[1..1]	
	DtmVariableReference	S	[1..1]	
StructuredElements	STRUCT			Collection of structured elements
	StructuredElement	M	[1..*]	
StructuredVariable	STRUCT			Describes a binary value and its structure information
	BinaryVariable	M	[1..1]	
	StructuredElements	M	[1..1]	
	dataTypeDescriptor	O	[0..1]	
SubstituteType	STRUCT			Type of a substitute value
	substituteType	M	[1..1]	
SubstituteValue	STRUCT			Describes a substitute value which is used in combination with the
	choice of	M	[1..1]	

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
	SubstituteType	S	[1..1]	behavior of disturbed channel values
	Variant	S	[1..1]	
SupportedDatasetFormat	STRUCT			A list of the supported data set that can be upgraded by the DTM
	dataSetID	M	[1..1]	
	levelOfSupport	O	[0..1]	
	description	O	[0..1]	
SupportedLanguages	STRUCT			Collection of language ids
	LanguageId	M	[1..*]	
TimeData	STRUCT			Element of a time date value
	time	M	[1..1]	
Unit	STRUCT			Current unit and the collection of possible units of a process variable
	readAccess	O	[0..1]	
	writeAccess	O	[0..1]	
	choice of	O	[0..1]	
	EnumeratorVariable	S	[1..1]	
	ChannelReference	S	[1..1]	
UpperRange	STRUCT			Definition of an upper range element
	choice of	O	[0..1]	
	NumberData	S	[1..1]	
	StringData	S	[1..1]	
	TimeData	S	[1..1]	
	ChannelReference	S	[1..1]	
UpperRawValue	STRUCT			A numeric entry which reflects the actual value transferred via fieldbus. The value is mapped to the related range (UpperRangeValue). For example, if a PROFIBUS device maps a range from 10 mbar to 100 mbar to an integer of value 1 024 to 4 096 the 'UpperRawValue' contains 4 096, the 'UpperRange' contains 4 096
	number	M	[1..1]	
Value	STRUCT			Contains the display string for a variable or the variable itself
	readAccess	O	[0..1]	
	writeAccess	O	[0..1]	
	choice of	M	[1..1]	
	Display	S	[1..1]	
	Variant	S	[1..1]	
Variable	STRUCT			Selected element of an enumeration
	EnumeratorEntry	M	[1..1]	

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
Variant	STRUCT			Variable with data type and display format
	dataType	M	[1..1]	
	byteLength	O	[0..1]	
	displayFormat	O	[0..1]	
	choice of	M	[1..1]	
	StringData	S	[1..1]	
	NumberData	S	[1..1]	
	TimeData	S	[1..1]	
	EnumeratorVariable	S	[1..1]	
	BitEnumeratorVariable	S	[1..1]	
	BinaryVariable	S	[1..1]	
	StructuredVariable	S	[1..1]	
VersionInformation	STRUCT			Description of the version of a component where used. For example – version of the DTM (if used in DtmInfo) – version of the physical device (if used in DtmDeviceType)
	readAccess	O	[0..1]	
	writeAccess	O	[0..1]	
	name	M	[1..1]	
	vendor	O	[0..1]	
	version	O	[0..1]	
	date	O	[0..1]	
	descriptor	O	[0..1]	

A.4 User information data types

Namespace: user

The simple data types (see Table A.5) and structured data types (Table A.6) defined in Clause A.4 are used as a base for the definition of service-specific data types or as service arguments.

Table A.5 – Simple user information data types

Data type	Definition	Description
administrator	BOOL	Flag describing if the user has administrative rights (default to "FALSE")
loginLocation	STRING	Description of the location the user logged into
loginTime	DATE_AND_TIME	Time of last login
oemService	BOOL	Flag describing if the user has OEM service rights (default to "FALSE")
projectName	STRING	Unique identifier for the project within the name space of the Frame Application
sessionDescription	STRING	Description of the user session (may be given by the user)
userLevel	enumeration (observer | operator | maintenance | planningEngineer)	User level specifying user's rights
userName	STRING	Name of the current user

Table A.6 – Structured user information data type

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
FDTUserInformation	STRUCT			Description of the user role including information about permissions, current session, etc.
	projectName	M	[1..1]	
	userName	M	[1..1]	
	userLevel	M	[1..1]	
	oemService	O	[0..1]	
	administrator	O	[0..1]	
	loginTime	O	[0..1]	
	loginLocation	O	[0..1]	
	sessionDescription	O	[0..1]	

A.5 DTM information data type

Namespace: dtmInfo

The data type (see Table A.7) defined in Clause A.5 is used as a base for the definition of service-specific data types or as service arguments.

Table A.7 – Structured DTM information data type

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
DtmInfo	STRUCT			Describes the DTM itself
	fdt:FDTVersion	M	[1..1]	
	fdt:VersionInformation	M	[1..1]	
	fdt:DtmDeviceTypes	M	[1..*]	

A.6 BTM data types

Namespace: btm

The simple data types (see Table A.8) and structured data types (see Table A.9) defined in Clause A.6 are used as a base for the definition of service-specific data types or as service arguments.

Table A.8 – Simple BTM data types

Data type	Definition	Description
blockCategory	UUID	Unique identifier for a block type such as Analog Input, Resource Block
blockCategoryName	STRING	Human readable block type name
blockTag	STRING	Block Tag is a unique identifier for the block
index	UDINT	An integer value that provides the offset from the beginning of the block or from the beginning of the structure
label	STRING	Readable name. If the optional attribute 'label' exists, the attribute 'name' contains a language independent identifier
profile	UINT	Number assigned by the fieldbus organization that uniquely identifies the profile on which the block is based
profileRevision	UINT	The revision number of the profile on which the block definition is based
slot	UDINT	An integer provides a parameter-grouping number. For example, in Profibus, it can provide the slot number (if the device supports slots), and in Foundation™ Fieldbus[4], it can provide the Application VFD number if the instrument supports more than one application VFD
usage	enumeration (contained | input | output | eventSource | eventSink)	An indication of whether this object may be set by another function block (I), provides a value to another function block (O), or only provides communications support (C) to devices other than function blocks – of type Visible String, 1 Octet

4 FOUNDATION™ Fieldbus is the trade name of the non-profit consortium Fieldbus Foundation. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the trade name holder or any of its products. Compliance does not require use of the trade name. Use of the trade name requires permission of the trade name holder.

Table A.9 – Structured BTM data types

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
BlockIcon	STRUCT			Icon for a Block
	fdt:path	M	[1..1]	
BlockTypeCategory	STRUCT			Type of the Block
	blockCategory	M	[1..1]	
	blockCategoryName	O	[0..1]	
BtmBlockType	STRUCT			Description of a block type
	fdt:readAccess	O	[0..1]	
	fdt:writeAccess	O	[0..1]	
	fdt:manufacturerId	O	[0..1]	
	profile	O	[0..1]	
	profileRevision	O	[0..1]	
	fdt:VersionInformation	M	[1..1]	
	fdt:SupportedLanguages	O	[0..1]	
	BlockTypeCategory	O	[0..1]	
	fdt:BusCategories	M	[1..1]	
	BlockIcon	O	[0..1]	
BtmVariable	STRUCT			Block variable description with name, index, value, range, etc.
	fdt:name	M	[1..1]	
	label	M	[1..1]	
	index	M	[1..1]	
	slot	O	[0..1]	
	fdt:descriptor	O	[0..1]	
	usage	M	[1..1]	
	fdt:Value	M	[1..1]	
	fdt:Unit	O	[0..1]	
	fdt:Ranges	O	[0..1]	
	fdt:statusFlag	O	[0..1]	
	fdt:StatusInformation	O	[0..1]	
BtmVariables	STRUCT			Collection of BTM variables
	fdt:name	M	[1..1]	
	label	M	[1..1]	
	index	M	[1..1]	
	collection of	M	[1..1]	
	BtmVariables		[0..*]	
	BtmVariable		[0..*]	

A.7 Device and Scan identification data types

Namespace: ident

The data types defined in Clause A.7 (see Table A.10 and Table A.11) are used as a base for the definition of service-specific data types or as service arguments.

Table A.10 – Simple device identification data types

Data type	Definition	Description
configuredState	enumeration (configuredAndPhysicallyAvailable | configuredAndNotPhysicallyAvailable | availableButNotConfigured | notApplicable)	The attribute is only used for protocols that have a master configuration such as Profibus. For all other protocols "notApplicable" is to be used. This attribute defines if a scanned module connected to this bus address is in a configured state or the attribute is optionally used by Frame Applications to display the information to the user. The following values are valid in the enumeration: – configuredAndPhysicallyAvailable – configuredAndNotPhysicallyAvailable – availableButNotConfigured – notApplicable
idDTMSupportLevel	enumeration (genericSupport | profileSupport | blockspecificProfileSupport | specificSupport | identSupport)	Defines the support level of a DTMDeviceType for a physical device. This attribute can be used by a Frame Application to display the support level of a DTM to the user. Users may use this information for an assignment decision. genericSupport: The DTMDeviceType applies to all kinds of physical devices of the corresponding fieldbus protocol. profileSupport: The DTMDeviceType applies to physical devices of a certain profile of the fieldbus protocol. blockspecificProfileSupport: The DTMDeviceType applies to blocks included in physical device types (e.g. Pressure TransducerBlock). specificSupport: The DTMDeviceType is developed for this physical device type. identSupport: The DTMDeviceType is capable of identifying the physical device in a vendor specific manner and to propose a better DTMDeviceType
match	STRING	Attribute contains a regular expression string, which shall match with an attribute provided by scan result
name	STRING	Contains the name of one single identification information
nomatch	STRING	Attribute contains a regular expression string, which shall not match with an attribute provided by the scan result
protocolSpecificName	STRING	Fieldbus protocol-specific name. This attribute provides a reference to the fieldbus specification
resultState	enumeration (provisional | final | error)	Marks the XML document as first provisional result provided by ScanResponse(). DTM will send further provisional XMLs (resultState = "provisional") and one result document at the end of the scan operation (resultState = "final"). In case of an error, the result indicates this by setting resultState = error
value	STRING	Attribute contains a regular expression string, which shall match with an attribute provided by scan result

Table A.11 – Structured device identification data types

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
DeviceTypeIdentification	STRUCT			Contains all identification elements of a device type for a physical device type or group
	idDTMSupportLevel	M	[1..1]	
	IdBusProtocol	M	[1..1]	
	IdBusProtocolVersion	M	[1..1]	
	IdManufacturer	M	[1..1]	
	IdTypeID	M	[1..1]	
	IdSoftwareRevision	M	[1..1]	
	IdHardwareRevision	M	[1..1]	
	IdValues	O	[0..1]	
DeviceIdentificationList	STRUCT			List of identifications of several device types for physical device types. The DeviceTypeIdentifications element should not be used in the context of data returned by service GetIdentificationInformation. The element should only be used in the context of data which could contain identification information for several device types (i.e. services Scan, HardwareInformation)
	DeviceIdentification	M	[1..*]	
IdAddress	STRUCT			Contains the fieldbus address of a device or a block address. This information is available in scan response only. It cannot be used for matching
	ident:value	M	[1..1]	
	ident:protocolSpecificName	M	[1..1]	
IdBusProtocol	STRUCT			Provides the protocol variant, e.g. Profibus PA
	value	O	[0..1]	
	protocolSpecificName	M	[1..1]	
	RegExpr	O	[0..*]	This element is not available in a Response of the service Scan
IdBusProtocolVersion	STRUCT			Contains the version of the protocol, e.g. 5 or 6 in case of HART®[5], or 3.0 in case of Profibus PA
	value	O	[0..1]	
	protocolSpecificName	M	[1..1]	
	RegExpr	O	[0..*]	This element is not available in a Response of the service Scan
IdDeviceTag	STRUCT			Contains the tag of the scanned device or block instance. This attribute is only available in scan result and for information only. It cannot be used for matching
	value	O	[0..1]	
	protocolSpecificName	M	[1..1]	
IdManufacturer	STRUCT			Identifies the manufacturer of the device or block
	value	O	[0..1]	
	protocolSpecificName	M	[1..1]	
	RegExpr	O	[0..*]	This element is not available in a Response of the service Scan

5 HART is the registered trade name of the HART Communication Foundation. This information is given for convenience of users of this document and does not constitute an endorsement by IEC of the trade name holder or any of its products. Compliance does not require use of the trade name. Use of the trade name requires permission of the trade name holder.

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
IdSerialNumber	STRUCT			Contains the serial number of a scanned device or block. This attribute is only available in scan online process and for information only. It cannot be used for matching. NOTE It is possible that a serial number is a unique identifier only in the context of one manufacturer or of one device type. In order to present a worldwide unique device identification, a Frame Application has to combine IdManufacturer, IdTypeID and IdSerialNumber
	value	O	[0..1]	
	protocolSpecificName	M	[1..1]	
IdTypeID	STRUCT			Identifies the device or block type
	value	O	[0..1]	
	protocolSpecificName	M	[1..1]	
	RegExpr	O	[0..*]	This element is not available in a Response of the service Scan
IdSoftwareRevision	STRUCT			Contains the software version of the device or block
	value	O	[0..1]	
	protocolSpecificName	M	[1..1]	
	RegExpr	O	[0..*]	This element is not available in a Response of the service Scan
IdHardwareRevision	STRUCT			Contains the hardware version of the device or block
	value	O	[0..1]	
	protocolSpecificName	M	[1..1]	
	RegExpr	O	[0..*]	This element is not available in a Response of the service Scan
IdDTMSupportLevel	STRUCT			This element is used only in DTMDeviceTypeIdentSchema and not in DTMScanIdentSchema. It cannot be used for matching. It can be displayed by Frame Applications applications for an assignment decision by the user
	value	O	[0..1]	
	protocolSpecificName	M	[1..1]	
	RegExpr	O	[0..*]	This element is not available in a Response of the service Scan
IdValue	STRUCT			Contains the name value pair for one identification parameter without the semantic information for the Frame Application
	name	M	[1..1]	
	value	O	[0..1]	
	protocolSpecificName	M	[1..1]	
	RegExpr	O	[0..*]	This element is not available in a Response of service Scan
IdValues	STRUCT			List of identification elements that are not specifically defined by Idxxx elements listed above. No further protocol-independent semantic information is available for those identification elements
	IdValue	O	[0..*]	

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
RegExpr	STRUCT			Contains one or more regular expression attributes (refer to the implementation-specific definition of regular expression for example in IEC TR 62453-41, IEC TR 62453-42, or IEC TR 62453-43) of type match or nomatch
	match	O	[0..1]	
	nomatch	O	[0..1]	
ScanIdentification	STRUCT			Contains all identification elements for one single scanned device or block. This includes elements with well defined semantics as well as IdValues
	configuredState	O	[0..1]	
	fdt:CommunicationError	O	[0..1]	
	IdBusProtocol	M	[1..1]	
	IdBusProtocolVersion	M	[1..1]	
	IdAddress	M	[1..1]	
	IdManufacturer	M	[1..1]	
	IdTypeID	M	[1..1]	
	IdSoftwareRevision	M	[1..1]	
	IdHardwareRevision	M	[1..1]	
	IdDeviceTag	M	[1..1]	
	IdSerialNumber	M	[1..1]	
	IdValues	O	[0..1]	
ScanIdentificationList	STRUCT			List of identifications for several scanned physical devices or blocks
	fdt:protocolId	M	[1..1]	
	resultState	M	[1..1]	
	ScanIdentification	O	[0..*]	

A.8 Function data types

Namespace: func

The simple data types (see Table A.12) and structured data types (see Table A.13) defined in Clause A.8 are used as a base for the definition of service-specific data types or as service arguments.

Table A.12 – Simple function data types

Data type	Definition	Description
checked	BOOL	Status of a menu entry
defaultFunctionId	DINT	Contains a function ID which can be used to open a default Presentation object or to raise a default function
enabled	BOOL	Status of a menu entry
hasGUI	BOOL	Specifies whether the DTM supports a user interface for an extended function
help	STRING	Human readable help string for a function
hidden	BOOL	Status of a menu entry
label	STRING	Human readable label of a function
mime-type	STRING	Mime-type of a document provided by a DTM
moduleId	UDINT	Unique identifier for a module within the name space of the device instance. The same ids as defined in the DTMParameter document shall be used
path	URI	Path to an object that has to be embedded for a function like bitmaps or other graphical elements
printable	BOOL	Specifies whether the DTM supports a printable document for a function (access via service GetDocumentation)
printableStandardFunction	BOOL	Specifies whether the standard function is printable
programName	STRING	Specifies the name of a document viewer program
resizable	BOOL	Specifies whether the GUI is resizable
resizableStandardFunction	BOOL	Specifies whether the GUI of a standard function is resizable
separator	BOOL	Special menu entry
toggle	BOOL	Status of a menu entry whether it is active or not

Table A.13 – Structured function data types

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
Document	STRUCT			Definition of a document which is provided by a DTM and displayed by the Frame Application, if a user selects the entry
	label	M	[1..1]	
	help	M	[1..1]	
	path	M	[1..1]	
	choice of	M	[1..1]	
	MimeType	S	[1..1]	
	OpenWith	S	[1..*]	
	Icon	O	[0..1]	
FDTFunctionCall	STRUCT			Definition of the element which specifies a specific function call. Contains fdt:FunctionID
	fdt:functionId	M	[1..1]	
	fdt:ApplicationId	O	[0..1]	
	ops:operationPhase	O	[0..1]	
Function	STRUCT			Definition of a single function entry which is not a standard function (concerning the ApplicationIds): Contains an fdt:FunctionID
	label	M	[1..1]	
	fdt:name	M	[1..1]	
	help	M	[1..1]	
	hasGUI	O	[0..1]	
	printable	O	[0..1]	
	resizable	O	[0..1]	
	functionId	M	[1..1]	
	Icon	O	[0..1]	
	Status	O	[0..1]	
	fdt:FDTApplicationIds	O	[0..1]	
Functions	STRUCT			Definition of a group of function entries
	label	M	[1..1]	
	fdt:name	M	[1..1]	
	help	M	[1..1]	
	moduleId	O	[0..1]	
	Status	O	[0..1]	
	collection of	O	[0..*]	
	Function		[0..*]	
	Document		[0..*]	
	StandardFunction		[0..*]	
	Functions		[0..*]	
	StandardFunctions		[0..*]	
GUIInformation	<technology specific>			Information about how to start a user interface of a function
Icon	STRUCT			Icon for a function
	path	M	[1..1]	
MimeType	STRUCT			Mime-type of a document provided by a DTM. The mime-type is used by the Frame Application to determine an appropriate viewer for a document
	mime-type	O	[0..1]	

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
OpenWith	STRUCT			Name of the program which should be used to open a document
	programName	O	[0..1]	
StandardFunction	STRUCT			Definition of a single function entry which is a standard function (concerning the applicationIds). This entry does not have its own label. The label shall be supplied by the Frame Application
	fdt:name	M	[1..1]	
	help	M	[1..1]	
	printableStandardFunction	O	[0..1]	
	resizableStandardFunction	O	[0..1]	
	functionId	M	[1..1]	
	Icon	O	[0..1]	
	Status	O	[0..1]	
	fdt:ApplicationId	M	[1..1]	
StandardFunctions	STRUCT			Definition of a group of standard function entries. This entry does not have its own label. The label shall be supplied by the Frame Application. It cannot contain standard functions
	fdt:ApplicationId	M	[1..1]	
	fdt:name	M	[1..1]	
	help	M	[1..1]	
	moduleId	O	[0..1]	
	Status	O	[0..1]	
	collection of	O	[0..*]	
	Function		[0..*]	
	Functions		[0..*]	
Status	STRUCT			Status of a function
	toggle	M	[1..1]	
	checked	M	[1..1]	
	enabled	M	[1..1]	
	hidden	M	[1..1]	
	separator	M	[1..1]	

A.9 AuditTrail data types

Namespace: auditTrail

The simple data types (see Table A.14) and structured data types (see Table A.15) defined in Clause A.9 are used as a base for the definition of service-specific data types or as service arguments.

Table A.14 – Simple auditTrail data types

Data type	Definition	Description
path	URI	Path to an object that has to be embedded within the document such as bitmaps or other graphical elements
title	STRING	Human readable title for the documentation

Table A.15 – Structured auditTrail data types

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
AuditTrailDeviceStatusEvent	STRUCT			Description of the status of a device
	fdt:statusFlag	M	[1..1]	
	fdt:StatusInformation	O	[0..1]	
LogEntry	STRUCT			Notification about changed variables, executed functions, or device status events
	fdt:descriptor	O	[0..1]	
	choice of	M	[1..1]	
	AuditTrailFunctionEvent	S	[1..1]	
	AuditTrailVariableEvent	S	[1..1]	
	AuditTrailDeviceStatus-Event	S	[1..1]	
AuditTrailFunctionEvent	STRUCT			Description about an executed function
	fdt:display	M	[1..1]	
AuditTrailVariable	STRUCT			Description of a changed variable
	fdt:display	M	[1..1]	
	fdt:Unit	O	[0..1]	
	fdt:Ranges	O	[0..1]	
	fdt:statusFlag	O	[0..1]	
	fdt:StatusInformation	O	[0..1]	
AuditTrailVariableEvent	STRUCT			Notification about a changed variable
	fdt:name	M	[1..1]	
	fdt:descriptor	O	[0..1]	
	ChangedFrom	M	[1..1]	
	ChangedTo	M	[1..1]	
ChangedFrom	STRUCT			Last value of an audit trail variable
	AuditTrailVariable	M	[1..1]	
ChangedTo	STRUCT			New value of an audit trail variable
	AuditTrailVariable	M	[1..1]	
TransactionInfo	STRUCT			Used to request start and stop audit trail for the following actions, e.g. configuration or simulation
	systemTag	M	[1..1]	

A.10 Documentation data types

Namespace: doc

The simple data types (see Table A.16) and structured data types (see Table A.17) defined in Clause A.10 are used as a base for the definition of service-specific data types or as service arguments.

Table A.16 – Simple documentation data types

Data type	Definition	Description
path	URI	Path to an object that has to be embedded within the document such as bitmaps or other graphical elements
title	STRING	Human readable title for the documentation

Table A.17 – Structured documentation data types

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
Documentation	STRUCT			Documentation of a DTM for a specific FDTFunctionCall
	title	M	[1..1]	
	fdt:classificationId	O	[0..1]	
	fdt:manufacturerId	O	[0..1]	
	fdt:deviceTypeId	O	[0..1]	
	dtmi:deviceTypeInformation	O	[0..1]	
	fdt:descriptor	O	[0..1]	
	fdt:date	O	[0..1]	
	fdt:windowTitle	O	[0..1]	
	fdt:VersionInformation	M	[1..1]	
	fdt:ClassificationIds	O	[0..1]	
	DocumentVariables	M	[1..1]	
	choice	O	[0..1]	
	DTMStyleForComplete-Document	M	[1..1]	
	DTMSpecificXMLData	M	[1..1]	
DocumentVariable	STRUCT			Human readable variable description with name, value, range, etc.
	fdt:name	M	[1..1]	
	fdt:descriptor	O	[0..1]	
	fdt:display	M	[1..1]	
	fdt:Unit	O	[0..1]	
	fdt:Ranges	O	[0..1]	
	fdt:statusFlag	O	[0..1]	
	fdt:StatusInformation	O	[0..1]	
DocumentVariables	STRUCT			Collection of document variables
	fdt:name	M	[1..1]	
	fdt:descriptor	O	[0..1]	
	collection of	M	[1..1]	
	DocumentVariables		[0..*]	
	DocumentVariable		[0..*]	
	GraphicReference		[0..*]	
DTMSpecificXMLData	STRUCT			Optional additional information which shall be described by a private style
DTMStyleForComplete Document	STRUCT			Optional style information which has to be provided by a DTM if it returns documents which cannot be described by the FDT standard style
GraphicReference	STRUCT			Reference to an object that has to be embedded within the document such as bitmaps or other graphical elements
	title	M	[1..1]	
	fdt:descriptor	O	[0..1]	
	path	M	[1..1]	

A.11 DeviceList data type

Namespace: devList

The simple data types (see Table A.18) and structured data types (see Table A.19) defined in Clause A.11 are used as a base for the definition of service-specific data types or as service arguments.

Table A.18 – Simple deviceList data type

Data type	Definition	Description
errorDescription	STRING	Detailed error information in case of dtmSpecificError
errorInfo	enumeration (ok | failedToSet | failedDuplicateAddress | cancelled | dtmSpecificError)	To be used when DTMDeviceListSchema is returned to indicate an error summary if used as an attribute of the DeviceList element and an error information for a specific address when used in the Device element. Enumeration: – "ok", address was set successfully – used in DeviceList as well as in Device – "failedToSet" – used in Device to indicate failed NetworkManagementInfoWrite – "failedDuplicateAddress" – used to indicate that the address is already available and could not be set – "cancelled" – used to indicate that the setting was cancelled by user – "dtmSpecificError" – indicates a DTM specific error. In this case, the error description text is to be used to give more detailed error information
showUserInterface	enumeration (openUserInterface | noUserInterface | setNextValidAddress)	Indicates if the Communication Channel should open a user interface in order to get a protocol-specific address selection by decision of the user. Enumeration: – openUserInterface – user interface should be opened to request the address from user – noUserInterface – no user interface should be opened – setNextValidAddress – Communication Channel has to set the next valid address without using a user interface. In this case, the busAddress is not used by the Communication Channel If this attribute is not set, NoUserInterface is assumed

Table A.19 – Structured deviceList data type

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
BusAddress	STRUCT			Contains a single busAddress element
	fdt:busAddress	M	[1..1]	
BusAddressRange	STRUCT			Bus address range. The structure data type defines a start and an end bus address (for example used in ScanRequest)
	BusAddress	M	[1..*]	
Device	STRUCT			Contains device identification information and system tag of the corresponding DTM
	fdt:systemTag	M	[1..1]	
	errorInfo	O	[0..1]	
	errorDescription	O	[0..1]	
	BusAddressRange	M	[1..1]	
DeviceList	STRUCT			List of DTM system tags and corresponding device addresses to set
	errorInfo	O	[0..1]	
	errorDescription	O	[0..1]	
	showUserInterface	O	[0..1]	
	Device	M	[1..*]	

A.12 Network management data types

Namespace: net

The simple data types (see Table A.20) and structured data types (see Table A.21) defined in Clause A.12 are used as a base for the definition of service-specific data types or as service arguments.

Table A.20 – Simple network management data types

Data type	Definition	Description
busAddress	String	Protocol-specific address of device
configurationData	ARRAY OF USINT	Protocol-specific configuration data as binary stream according to the fieldbus specification
moduleId	UDINT	Unique identifier for a module within the name space of the device instance
moduleTypeId	UDINT	Unique identifier for a module type within the name space of the device type
redundant	BOOL	Specifies whether a device or module is redundant
slot	UDINT	Unique identifier for the slot of a module within the name space of the device instance

Table A.21 – Structured network management data types

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
DeviceAddress	STRUCT			Protocol-specific address of device
	fdt:busAddress	M	[1..1]	
	configurationData	O	[0..1]	
UserDefinedBus	STRUCT			Protocol-specific part of NetworkInfo, is defined within the IEC 62453-3xy documents
	<protocol specific>			
NetworkInfo	STRUCT			Description of network configuration of a device instance
	fdt:protocolId	M	[1..1]	
	fdt:BusRedundancy	O	[0..1]	
	choice of	O	[0..*]	
	DeviceAddress	S	[1..1]	
	UserDefinedBus	S	[1..1]	
DtmDeviceInstanceTopology	STRUCT			Description of internal topology of a DtmDeviceType
	fdt:readAccess	O	[0..1]	
	fdt:writeAccess	O	[0..1]	
	NetworkInfo	O	[0..1]	
	fdt:ChannelReferences	O	[0..1]	
	InternalChannel	M	[1..*]	
InternalChannel	STRUCT			An internal channel is the connection point for an internal module within the internal topology
	fdt:readAccess	O	[0..1]	
	fdt:writeAccess	O	[0..1]	
	Module	O	[0..*]	
Module	STRUCT			Description of a hardware or software module of a device
	fdt:readAccess	O	[0..1]	
	fdt:writeAccess	O	[0..1]	
	moduleId	M	[1..1]	
	moduleTypeId	O	[0..1]	
	slot	O	[0..1]	
	redundant	O	[0..1]	
	configurationData	O	[0..1]	
	fdt:VersionInformation	M	[1..1]	
	fdt:ChannelReferences	O	[0..1]	

A.13 Instance data types

Namespace: param

The simple data types (see Table A.22) and structured data types (see Table A.23) defined in Clause A.13 are used as a base for the definition of service-specific data types or as service arguments.

Table A.22 – Simple instance data types

Data type	Definition	Description
itemErrorDescription	enumeration (dtmSpecific | noLock | notLongerValid | outOfResources | invalidValue)	Enumeration describing an error: – dtmSpecific – DTM-specific error – noLock – instance data set could not be locked – notLongerValid – the item is not longer valid. This may happen due to configuration changes – outOfResources – the DTM has no resources to perform the request. This may happen if the DTM cannot queue the request – invalidValue – the requested value is invalid
itemId	STRING	Unique id of an item
itemKind	enumeration (alarm | analogInput | analogOutput | computation | contained | correction | device | diagnostic | digitalInput | digitalOutput | discrete | discreteInput | discreteOutput | dynamic | frequency | frequencyInput | frequencyOutput | hart | input | local | localDisplay | operate | output | sensorCorrection | service | tune |others)	Identification of the item context. This is some kind of classification regarding the type of value. Enumeration: – alarm – contains alarm limits – analogInput – is part of an analog input block – analogOutput – is part of an analog output block – computation – is part of a computation block – contained – represents the physical characteristics of the device – correction – is part of the correction block – device – represents the physical characteristics of the device – diagnostic – indicates the device status – digitalInput – is part of a digital input block – digitalOutput – is part of a digital output block – discrete – is part of a discrete block – discreteInput – is part of a discrete input block – discreteOutput – is part of a discrete output block – dynamic – is modified by the device without stimulus from the network – frequency – is part of frequency block – frequencyInput – is part of a frequency input block – frequencyOutput – is part of a frequency output block – hart – is part of HART block – input – is part of an input block. An input block is a special kind of computation block which does unit conversions, scaling, and damping. The parameter of the input block parameters can be determined by the output of another block – local – is locally used by an application. Local items are not stored in a device, but they can be sent to a device – localDisplay – is part of the local display block. A local display block contains the items associated with the local interface (keyboard, display, etc.) of the device – operate – is used to control a block's operation output, is part of the output block. The values of output items may be accessed by another block input – sensorCorrection – is part of the sensor correction block – service – is used when performing routine maintenance – tune – is used to tune the algorithm of a block – others – is used if all other entries do not match
itemType	enumeration (standard | specific)	Information on whether the item follows the semantics defined via the general rules defined for a specific protocol (standard) or a DTM/vendor specific semantics (specific)

Data type	Definition	Description
label	STRING	Human readable name
limitBits	enumeration (none | low | high | constant)	Limit status of the item. NOTE Additional information can be found in the OPC XML-DA Specification Version 1.0
moduleName	STRING	This attribute contains the name of the module
qualityBits	enumeration (bad | badConfigurationError | badNotConnected | badDeviceFailure | badSensorFailure | badLastKnownValue | badCommFailure | badOutOfService | badWaitingForInitialData | uncertain | uncertainLastUsableValue | uncertainSensorNotAccurate | uncertainEUExceeded | uncertainSubNormal | good | goodLocalOverride)	Quality status of the item. NOTE Additional information can be found in the OPC XML-DA Specification Version 1.0

Table A.23 – Structured instance data types

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
DtmItem	STRUCT			Contains the value of a parameter or a process value and some additional optional information such as time stamp
	fdt:id	M	[1..1]	
	TimeStamp	M	[1..1]	
	Quality	M	[1..1]	
	choice of	M	[1..1]	
	fdt:Variant	S	[1..1]	
	ItemError	S	[1..1]	
DtmItemInfo	STRUCT			Describes a parameter or a process value that is available via the Services to access the instance data of a DTM. The information contains descriptive attributes such as name as well as information on how the item is accessible. The relation between the item information and the item itself is realized via ItemId
	fdt:id	M	[1..1]	
	fdt:SemanticInformation	M	[1..*]	The DTM shall provide a <SemanticInformation> element for all supported fieldbus protocols of the DTM instance
	fdt:name	M	[1..1]	
	fdt:dataType	M	[1..1]	
	itemType	M	[1..1]	
	fdt:descriptor	O	[0..1]	
	moduleName	O	[0..1]	
	fdt:readAccess	O	[0..1]	
	fdt:writeAccess	O	[0..1]	
	label	O	[0..1]	
	ItemKind	M	[1..*]	
	UnitDescription	O	[0..1]	
	choice of	M	[1..1]	
	RangeDescriptions	S	[0..1]	
	ValueDescription	S	[0..1]	
DtmItemInfoGroup	STRUCT			List of DTM item information
	fdt:name	M	[1..1]	
	fdt:semanticId	M	[1..1]	
	label	O	[0..1]	
	DtmItemInfo	O	[0..*]	
	DtmItemInfoGroup	O	[0..*]	
DtmItemInfoList	STRUCT			List of DTM item information and/or a list of item information groups
	DtmItemInfo	O	[0..*]	
	DtmItemInfoGroup	O	[0..*]	
DtmItemList	STRUCT			List of DTM items
	DtmItem	M	[1..*]	
DtmItemSelection	STRUCT			Items selected for execution of a service (or similar)
	fdt:id	M	[1..1]	

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
DtmItemSelectionList	STRUCT			List of selected items
	DtmItemSelection	M	[1..*]	
ItemError	STRUCT			Communication error or other error
	choice of	O	[0..1]	
	ItemErrorDescription	S	[1..1]	
	fdt:CommunicationError	S	[1..1]	
ItemErrorDescription	STRUCT			Description of non-communication error
	itemErrorDescription	M	[1..1]	
	fdt:descriptor	O	[0..1]	
ItemKind	STRUCT			Identification of the item context
	itemKind	M	[1..1]	
ItemReference	STRUCT			Reference to another item within the XML document
	fdt:idref	O	[0..1]	
LowerRange-Description	STRUCT			Description of the lower range
	choice of	O	[0..1]	
	ItemReference	S	[1..1]	
	fdt:StringData	S	[1..1]	
	fdt:NumberData	S	[1..1]	
	fdt:TimeData	S	[1..1]	
PossibleEnumera-tions	STRUCT			Possible enumerations of an item
	fdt:EnumeratorEntries	M	[1..1]	
Quality	STRUCT			Description of the quality of the item. For write requests: The Frame Application may define a Quality. If the underlying device/parameter supports the quality definitions, the value and quality should be handed over by the DTM to the device, if not the DTM should only process values with good quality
	qualityBits	M	[1..1]	
	limitBits	O	[0..1]	
RangeDescription	STRUCT			Description of a range
	LowerRangeDescription	M	[1..1]	
	UpperRangeDescription	M	[1..1]	
	fdt:LowerRawValue	O	[0..1]	
	fdt:UpperRawValue	O	[0..1]	
RangeDescriptions	STRUCT			Description of the ranges of the item
	RangeDescription	M	[1..*]	
TimeStamp	STRUCT			Description of the time stamp of the item
	fdt:time	M	[1..1]	
UnitDescription	STRUCT			Description of the unit of the item
	choice of	O	[0..1]	
	ItemReference	S	[1..1]	
	fdt:EnumeratorEntry	S	[1..1]	

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
UpperRangeDescription	STRUCT			Description of the upper range
	choice of	O	[0..1]	
	ItemReference	S	[1..1]	
	fdt:StringData	S	[1..1]	
	fdt:NumberData	S	[1..1]	
	fdt:TimeData	S	[1..1]	
ValueDescription	STRUCT			Description of the value of the item
	PossibleEnumerations	M	[1..1]	

A.14 DeviceStatus data types

Namespace: status

The simple data types (see Table A.24) and structured data types (see Table A.25) defined in Clause A.14 are used as a base for the definition of service-specific data types or as service arguments.

Table A.24 – Simple device status data types

Data type	Definition	Description
deviceInitiatedCommunication	BOOL	Device is currently using the device-initiated communication

Table A.25 – Structured device status data types

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
DtmDeviceStatus	STRUCT			Description of the current status of a device
	fdt:statusFlag	M	[1..1]	
	deviceInitiatedCommunication	O	[0..1]	
	fdt:StatusInformation	O	[0..1]	

A.15 OnlineCompare data types

Namespace: onlineComp

The simple data types (see Table A.26) and structured data types (see Table A.27) defined in Clause A.15 are used as a base for the definition of service-specific data types or as service arguments.

Table A.26 – Simple online compare data types

Data type	Definition	Description
statusFlag	enumeration (equal | notEqual | noComparableData)	Describes whether the data is equal or not

Table A.27 – Structured online compare data types

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
DTMOnlineCompare	STRUCT			Contains the compare result or a communication error
	statusFlag	M	[1..1]	
	fdt:StatusInformation	O	[0..1]	

A.16 UserInterface data types

Namespace: ui

The simple data types (see Table A.28) and structured data types (see Table A.29) defined in Clause A.16 are used as a base for the definition of service-specific data types or as service arguments.

Table A.28 – Simple user interface data types

Data type	Definition	Description
helpContext	INT	Help context (e.g. the reference number within the help file)
helpFile	STRING	Definition of the help file
messageButtons	enumeration (buttonsAbortRetryIgnore | buttonsOk | buttonsOkCancel | buttonsRetryCancel | buttonsYesNo | buttonsYesNoCancel)	Definition of the button types which shall appear within the message box
messageDefault	enumeration (buttonAbort | buttonRetry | buttonIgnore | buttonOk | buttonCancel | buttonYes | buttonNo)	Definition of the default button
messageType	enumeration (messageExclamation | messageInformation | messageQuestion | messageStop)	Type of a message such as exclamation, information, etc.
resultMessage	enumeration (nobutton | buttonAbort | buttonRetry | buttonIgnore | buttonOk | buttonCancel | buttonYes | buttonNo)	Definition of the result of the user interaction
resultStatus	enumeration (notSupported | denied | systemResponse | ok)	
runAsModal	BOOL	User interface as modal dialog
title	STRING	

Table A.29 – Structured user interface data types

Data type	Definition			Description
	Elementary data types	Usage	Multiplicity	
TextLine	STRUCT			Definition of a single text line within the message
	fdt:string	M	[1..1]	
UserMessage	STRUCT			Definition of the whole message
	runAsModal	O	[0..1]	
	messageType	M	[1..1]	
	messageButtons	M	[1..1]	
	messageDefault	M	[1..1]	
	title	M	[1..1]	
	helpFile	O	[0..1]	
	helpContext	O	[0..1]	
	collection of	M	[1..1]	
	TextLine		[0..*]	
	fdt:DtmVariable		[0..*]	
	resultMessage	O	[0..1]	
	resultStatus	O	[0..1]	

A.17 Fieldbus-specific data types

Table A.30 shows protocol-specific data types which shall be defined within an IEC 62453-3xy document describing protocol profile integration in FDT.

This namespace defines abstract data types that will be replaced by specific data types within the IEC 62453-3xy documents.

Namespace: fieldbus

Table A.30 – Fieldbus data types

Data type	Description
ChannelData	Protocol-specific identification information for a channel (see 7.6.1.2)
ConnectRequest	Protocol-specific information which is provided within a service Connect (see 7.6.1.2)
ConnectResponse	Protocol-specific information which is provided within a service Connect (see 7.6.1.2)
DeviceTypeIdentification	Protocol-specific identification information which can be transferred into devIdent:DeviceTypeIdentification
DeviceTypeIdentifications	Collection of DeviceTypeIdentification elements
DisconnectRequest	Protocol-specific information which is provided within a service Disconnect (see 7.6.1.3)
DisconnectResponse	Protocol-specific information which is provided within a service Disconnect (see 7.6.1.3)
ScanIdentification	Protocol-specific identification information which can be transferred into devIdent:ScanIdentification
ScanIdentifications	Collection of ScanIdentification elements
SequenceDefine	Protocol-specific information which is provided within a service SequenceBegin (see 7.6.1.7). A SequenceDefine data type typically contains multiple TransactionRequests and may contain additional protocol-specific information regarding the sequence execution
TransactionRequest	Protocol-specific information which is provided within a service Transaction (see 7.6.1.6)
TransactionResponse	Protocol-specific information which is provided within a service Transaction (see 7.6.1.6)

Bibliography

IEC TR 62453-41, *Field device tool (FDT) interface specification – Part 41: Object model integration profile – Common object model*

IEC TR 62453-42[6], *Field device tool (FDT) interface specification – Part 42: Object model integration profile – Common Language Infrastructure*

IEC TR 62453-43[7], *Field device tool (FDT) interface specification – Part 43: Object model integration profile – CLI and HTML*

ISO/IEC 19501:2005, *Information technology – Open Distributed Processing – Unified Modeling Language (UML) Version 1.4.2*

OPC XML-DA, *Specification Version 1.0*

VDI/VDE 2187, *Uniform user interface for digital field instruments*

6 Second edition under preparation. Stage at the time of publication: IEC DTR 62453-42:2022.

7 Under consideration.

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

SPÉCIFICATION DES INTERFACES DES OUTILS DES DISPOSITIFS DE TERRAIN (FDT) –

Partie 2: Concepts et description détaillée

AVANT-PROPOS

1) La Commission Électrotechnique Internationale (IEC) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de l'IEC). L'IEC a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. À cet effet, l'IEC – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de l'IEC"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec l'IEC, participent également aux travaux. L'IEC collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de l'IEC concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de l'IEC intéressés sont représentés dans chaque comité d'études.
3) Les Publications de l'IEC se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de l'IEC. Tous les efforts raisonnables sont entrepris afin que l'IEC s'assure de l'exactitude du contenu technique de ses publications; l'IEC ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de l'IEC s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de l'IEC dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de l'IEC et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) L'IEC elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de l'IEC. L'IEC n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à l'IEC, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de l'IEC, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de l'IEC ou de toute autre Publication de l'IEC, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de l'IEC peuvent faire l'objet de droits de brevet. L'IEC ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets.

L'IEC 62453-2 a été établie par le sous-comité 65E: Les dispositifs et leur intégration dans les systèmes de l'entreprise, du comité d'études 65 de l'IEC: Mesure, commande et automation dans les processus industriels. Il s'agit d'une Norme internationale.

Cette troisième édition annule et remplace la deuxième édition parue en 2016. Cette édition constitue une révision technique.

Cette édition inclut les modifications techniques majeures suivantes par rapport à l'édition précédente:

a) clarification relative à la fonction statique;
b) clarification relative à l'étiquette de l'interface utilisateur graphique système;
c) clarification relative à la perte de connexion.

Le texte de cette Norme internationale est issu des documents suivants:

Projet	Rapport de vote
65E/906/FDIS	65E/933/RVD

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à son approbation.

La langue employée pour l'élaboration de cette Norme internationale est l'anglais.

Le présent document a été rédigé selon les Directives ISO/IEC, Partie 2, il a été développé selon les Directives ISO/IEC, Partie 1 et les Directives ISO/IEC, Supplément IEC, disponibles sous www.iec.ch/members_experts/refdocs. Les principaux types de documents développés par l'IEC sont décrits plus en détail sous www.iec.ch/standardsdev/publications.

Une liste de toutes les parties de la série IEC 62453, publiées sous le titre général *Spécification des interfaces des outils des dispositifs de terrain (FDT)*, peut être consultée sur le site web de l'IEC.

Le comité a décidé que le contenu du présent document ne sera pas modifié avant la date de stabilité indiquée sur le site web de l'IEC sous webstore.iec.ch dans les données relatives au document recherché. À cette date, le document sera

- reconduit,
- supprimé,
- remplacé par une édition révisée, ou
- amendé.

IMPORTANT – Le logo "colour inside" qui se trouve sur la page de couverture de ce document indique qu'il contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer ce document en utilisant une imprimante couleur.

INTRODUCTION

La présente partie de l'IEC 62453 fournit une spécification d'interface pour les développeurs des composants des outils des dispositifs de terrain[1] (FDT - Field Device Tool) afin de prendre en charge le contrôle de fonction et l'accès aux données dans une architecture client/serveur. La spécification résulte d'un processus d'analyse et de conception destiné à réaliser des interfaces normalisées et permettre ainsi à de nombreux fournisseurs de développer des serveurs et des clients dans le cadre d'une interaction ininterrompue répondant à leur besoin.

L'intégration de bus de terrain dans les systèmes de commande nécessite d'effectuer quelques tâches supplémentaires. Outre les outils spécifiques aux bus de terrain et aux dispositifs, l'intégration de ces outils dans des outils d'ingénierie ou de planification à l'échelle d'un système de plus haut niveau s'avère nécessaire. La définition claire des interfaces d'ingénierie faciles à utiliser pour tous les outils concernés revêt une grande importance, en particulier pour une utilisation dans des systèmes de commande importants et hétérogènes, généralement dans le domaine de l'industrie de transformation.

Un composant logiciel spécifique à un équipement conçu conformément au présent document est appelé gestionnaire de Type d'Équipement (DTM - Device Type Manager). Il intègre toutes les données, fonctions et règles métier spécifiques à l'équipement dans le système par le biais des services FDT définis dans le présent document.

L'approche FDT/DTM s'applique à tous les types de bus de terrain et permet l'intégration d'une variété d'équipements dans des systèmes hétérogènes.

La Figure 1 représente l'alignement du présent document dans la structure de la série IEC 62453.

[1] FDT® est une marque de produits distribués par FDT Group AISBL. Cette information est donnée à l'intention des utilisateurs du présent document et ne signifie nullement que l'IEC approuve ou recommande l'emploi exclusif du produit cité. Des produits équivalents peuvent être utilisés s'il est démontré qu'ils conduisent aux mêmes résultats.

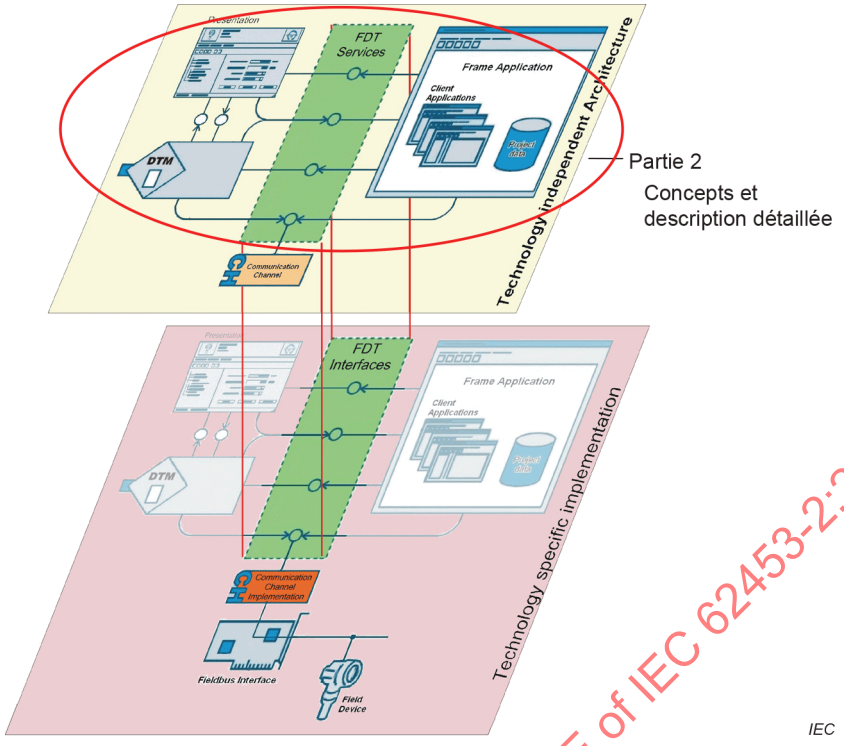


Figure 1 – Partie 2 de la série IEC 62453

SPÉCIFICATION DES INTERFACES DES OUTILS DES DISPOSITIFS DE TERRAIN (FDT) –

Partie 2: Concepts et description détaillée

1 Domaine d'application

La présente partie de l'IEC 62453 explique les principes courants du concept relatif aux outils des dispositifs de terrain. Ces principes peuvent être utilisés dans plusieurs applications industrielles telles que les systèmes d'ingénierie, les programmes de configuration et les applications de surveillance et de diagnostic.

Le présent document spécifie les objets généraux, leur comportement ainsi que leurs interactions qui constituent la base des FDT.

2 Références normatives

Les documents suivants sont cités dans le texte de sorte qu'ils constituent, pour tout ou partie de leur contenu, des exigences du présent document. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

IEC 61131 (toutes les parties), *Automates programmables*

IEC TR 62390:2005, *Common automation device – Profile guideline* (disponible en anglais seulement)

IEC 62453-1:2016, *Spécification des interfaces des outils des dispositifs de terrain (FDT) – Partie 1: Vue d'ensemble et guide*

IEC 62453-3xy (toutes les parties), *Spécification des interfaces des outils des dispositifs de terrain (FDT) – Partie 3xy: Intégration des profils de communication*

3 Termes, définitions, symboles, termes abrégés et conventions

3.1 Termes et définitions

Pour les besoins du présent document, les termes et définitions de l'IEC 62453-1 ainsi que les suivants s'appliquent.

L'ISO et l'IEC tiennent à jour des bases de données terminologiques destinées à être utilisées en normalisation, consultables aux adresses suivantes:

- IEC Electropedia: disponible à l'adresse http://www.electropedia.org/
- ISO Online browsing platform: disponible à l'adresse http://www.iso.org/obp

3.1.1
DTM Enfant
instance DTM dans un projet FDT, classée selon sa relation avec un DTM Parent

Note 1 à l'article: Tout DTM qui utilise une communication FDT peut être classé comme DTM Enfant (c'est-à-dire DTM d'Équipement, DTM de Passerelle, DTM de Module et BTM).

Note 2 à l'article: L'abréviation "DTM" est dérivée du terme anglais développé correspondant "Device Type Manager".

3.1.2
version du FDT
version de mise en œuvre définie par l'organisation spécifique à la technologie associée

Note 1 à l'article: La version du FDT est spécifiée dans l'IEC TR 62453-41 ou dans l'IEC TR 62453-42.

Note 2 à l'article: L'abréviation "FDT" est dérivée du terme anglais développé correspondant "Field Device Tool".

3.1.3
DTM monolithique
DTM unique représentant l'équipement complet avec tous ses modules

Note 1 à l'article: Il existe également d'autres concepts représentant les modules d'un équipement dans la présente spécification, par exemple le DTM de Module et le Gestionnaire de Type de Blocs (BTM - Block Type Manager).

Note 2 à l'article: L'abréviation "DTM" est dérivée du terme anglais développé correspondant "Device Type Manager".

3.1.4
DTM Parent
instance DTM dans un projet FDT, classée selon sa relation avec un DTM Enfant

Note 1 à l'article: Tout DTM qui assure une communication FDT peut être classé comme DTM Parent (par exemple, DTM de Communication, DTM de Passerelle, DTM d'Équipement composite).

Note 2 à l'article: L'abréviation "DTM" est dérivée du terme anglais développé correspondant "Device Type Manager".

3.2 Symboles et termes abrégés

Pour les besoins du présent document, les symboles et termes abrégés donnés dans l'IEC 62453-1 ainsi que les suivants s'appliquent.

DD	Device description (Description d'équipement)
OODMS	Object Oriented Database Management System (Système de Gestion de Bases de données orientées objet)
RDBMS	Relational Database Management System (Système de gestion de Bases de Données Relationnelles)

3.3 Conventions

3.3.1 Utilisation du langage UML

L'IEC 62453-1 définit les conventions de notation UML utilisées dans le présent document.

3.3.2 Déclaration de disponibilité d'état

La déclaration de disponibilité d'état utilise [_] et [X] pour déclarer la disponibilité d'un service dans un état particulier. Les expressions ont la signification suivante:

[_] '<nom de l'état>' ('<name of state>')	service indisponible dans cet état;
[X] '<nom de l'état>' ' ('<name of state>')	service disponible dans cet état.

3.3.3 Noms de types de données et références aux types de données

Les conventions pour la dénomination et le référencement des types de données sont expliqués à l'Article A.1.

4 Principes de base

4.1 Généralités

L'Article 4 présente le modèle des FDT et traite des sujets spécifiques aux exigences relatives à l'intégration des dispositifs de terrain.

4.2 Modèle abstrait des FDT

4.2.1 Vue d'ensemble du modèle des FDT

Le diagramme de classe UML représenté à la Figure 2 donne une vue d'ensemble des objets définis dans les FDT ainsi que leurs relations.

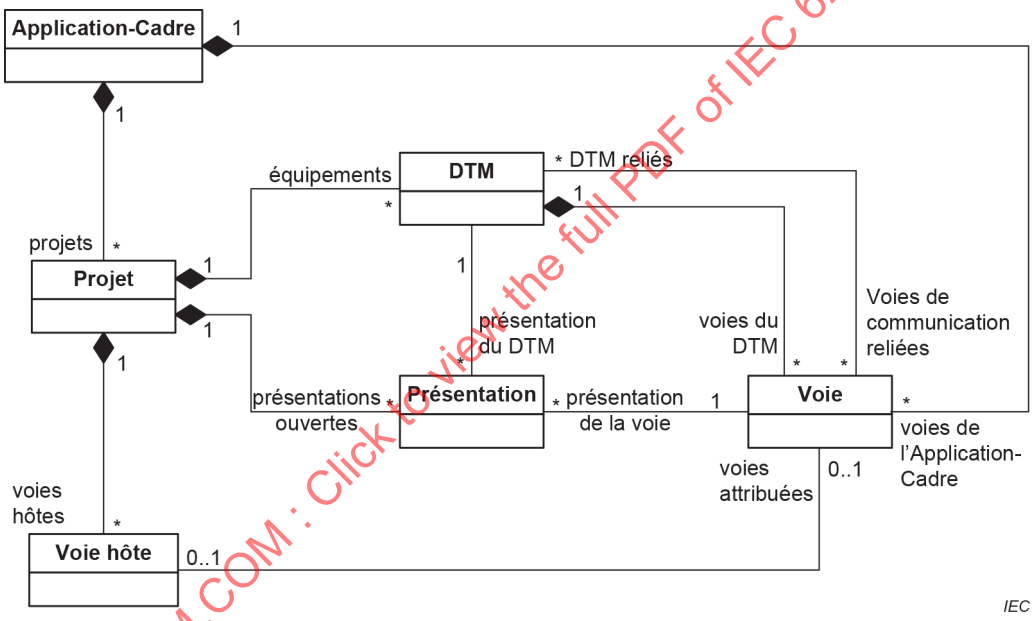


Figure 2 – Modèle abstrait des FDT

Le Tableau 1 comporte des descriptions succinctes de tous les objets. Une description plus détaillée des objets et des services associés est fournie de 4.2.2 à 4.2.4.

Les objets des FDT peuvent être tous exécutés sur une plate-forme ou dans un système distribué.

Les types de données définis dans l'Annexe A sont utilisés pour l'échange de données et l'interaction entre les objets. La mise en œuvre concrète de ces types de données est définie dans les documents relatifs au profil d'intégration des modèles d'ojets de la série IEC 62453 (l'IEC TR 62453-41, l'IEC TR 62453-42, par exemple) qui décrivent le mapping avec des technologies spécifiques de mise en œuvre.

Tableau 1 – Description des objets des FDT

Objet	Description
Application-Cadre	L'Application-Cadre désigne un objet logique pour représenter un environnement tel qu'un système d'ingénierie ou un outil autonome (voir 4.2.2). Elle contrôle la durée de vie des instances du DTM au sein du Projet. Une Application-Cadre peut prendre en charge plusieurs Projets.
Projet	Le Projet appartient à l'Application-Cadre. Le Projet désigne un objet logique décrivant la gestion des instances de l'équipement. Il contrôle la durée de vie ainsi que l'ensemble de données d'instance de l'équipement au sein d'une Application-Cadre. Les FDT ne définissent pas les services pour le Projet, car il s'agit d'un objet interne d'origine de l'Application-Cadre.
Voie Hôte	Une Voie Hôte appartient à l'Application-Cadre. La Voie Hôte désigne un objet logique représentant une partie d'une fonction d'un système hôte, comme DCS ou PLC. Elle est exigée lorsque des informations supplémentaires relatives au traitement des données E/S d'un équipement sont nécessaires, par exemple si des données du DTM ou du BTM font référence à plus d'une valeur E/S. Par exemple, un Octet est souvent utilisé afin de grouper les valeurs d'état de huit points E/S numériques. Dans ces applications, l'objet Voie Hôte fournit une association entre les contenus des données de la voie et les points E/S du processus individuel. Pour activer les données cycliques E/S au sein d'une Application-Cadre, une association doit exister entre les blocs fonctionnels E/S et les valeurs de processus d'un équipement. Les entrées et sorties des blocs fonctionnels E/S sont représentées par les Voies Hôtes. La valeur de processus est représentée par une Voie. L'association d'une Voie Hôte et d'une Voie est appelée attribution de la voie. Les FDT ne définissent pas les services pour la Voie Hôte, car il s'agit d'un objet interne d'origine de l'Application-Cadre
DTM	Un DTM désigne un composant logiciel comportant le logiciel d'application spécifique à un équipement. Il comprend toutes les données, fonctions et règles métier spécifiques à un équipement. Un DTM n'est généralement pas exécutable de manière autonome. Il nécessite une Application-Cadre (voir 4.2.2) qui agit comme l'environnement d'exécution. Un DTM est généralement développé par le fabricant d'équipements et expédié avec les équipements. Cela dépend de la conception logicielle du DTM, à savoir s'il peut être ou non utilisé pour un type d'équipement spécifique ou un groupe de types d'équipements différents, voire une famille de types d'équipements. Chaque équipement mis en place dans une installation industrielle est représenté par un objet DTM. Par conséquent, un ou plusieurs objets DTM sont nécessaires pour gérer les différents équipements. Les DTM sont installés dans le système afin que ce dernier puisse être dynamiquement agrandi en installant de nouveaux DTM pour de nouveaux équipements. Un DTM peut représenter différents types d'équipements, comme des dispositifs de terrain, des équipements de communication ou des équipements de passerelle (voir 4.2.3).
Présentation	Les objets Présentation (voir 4.2.3.3) représentent une interface utilisateur graphique (IUG). L'objet Présentation appartient au DTM, au BTM ou à la Voie.
Voie	L'objet Voie peut se comporter de trois manières différentes: Comme une Voie de Communication ('Communication Channe'), comme une Voie de processus ('Process Channel') et comme une combinaison des deux (voir 4.2.4).

Toutes les associations représentées à la Figure 2 sont décrites dans le Tableau 2.

Tableau 2 – Description des associations entre les objets des FDT

Association	Description
voies attribuées	Pour activer les données cycliques E/S au sein d'une Application-Cadre, une association doit exister entre les blocs fonctionnels E/S et les valeurs de processus d'un équipement. Les blocs fonctionnels E/S sont représentés par une Voie Hôte, et la valeur de processus par une voie. L'Application-Cadre (Projet) est responsable de cette association. Cas d'utilisation: – attribution de la voie Services: Services informationnels – service du DTM: GetChannels – service de la voie: ReadChannelInformation Services de gestion – service de la voie: WriteChannelInformation Services dépendants – services du DTM propriétaire pour la gestion de la voie
présentation de la voie	Les instances des objets Présentation sont liées à l'instance de leur objet métier DTM par cette relation. Cas d'utilisation: – tous les cas d'utilisation de la voie avec l'interface utilisateur graphique spécifique au DTM Services: Services informationnels (voir également 7.3) – service de la voie: GetChannelFunctions – service de la voie: GetGuiInformation Services dépendants – service de l'Application-Cadre: OpenPresentationRequest – service de l'Application-Cadre: ClosePresentationRequest
équipements	Un Projet détient la liste des instances du DTM par les équipements des associations. La publication du Projet entraîne la publication de toutes les instances du DTM associées. Une instance de DTM ne peut pas faire partie de plus d'un Projet. Cas d'utilisation: – génération du système – planification du système Services: Services informationnels – service de l'Application-Cadre: GetChildNodes Services de gestion – services relatifs à la gestion de la sous-topologie, voir 7.6.2 – ValidateAddChild – ChildAdded – ValidateRemoveChild – ChildRemoved – services de l'Application-Cadre associés à la gestion de la topologie
voies du DTM	Un DTM peut comporter une liste de voies, reliées par les voies du DTM. Une voie fait partie d'un DTM. Services: Services informationnels – service de l'Application-Cadre: GetChannels Services de gestion – Il n'existe aucun service pour gérer cette association. La durée de vie des instances des voies est contrôlée par un DTM

Association	Description
présentation du DTM	Les instances de l'objet Présentation sont associées à l'instance de leur DTM par cette relation. Cas d'utilisation: – tous les cas d'utilisation du DTM avec l'interface utilisateur graphique spécifique au DTM Services: – ouverture de la présentation – fermeture de la présentation
voies de l'Application-Cadre	L'Application-Cadre (projet) peut comporter une liste des Voies de Communication. Il n'existe aucun service pour gérer cette association. Voir 4.2.2.
Voies Hôtes	L'Application-Cadre met en correspondance les Voies de Processus des DTM avec la gestion E/S interne des projets.
Voies de Communication reliées	La topologie des DTM est représentée avec cette association. Dans l'arbre topologique, un objet DTM est le nœud enfant d'une voie. L'association est marquée par la connexion d'un équipement à une voie. L'Application-Cadre (projet) est responsable de cette association. Services: – les voies reliées peuvent être examinées avec GetParentNodes – SetLinkedCommunicationChannel – ReleaseLinkedCommunicationChannel
DTM reliés	La topologie des dispositifs de terrain est représentée avec cette association. Dans l'arbre topologique, un objet Voie est le nœud parent d'un équipement. L'association est marquée par la connexion d'une voie à un équipement. L'Application-Cadre (Projet) est responsable de cette association. Services: – les DTM reliés peuvent être examinés avec GetChildNodes – SetLinkedCommunicationChannel – ReleaseLinkedCommunicationChannel
présentations ouvertes	Les objets Présentations ouvertes (Opened presentations) sont reliés aux Projets par cette association. L'Application-Cadre (Projet) est responsable de cette association.
projets	Une Application-Cadre peut établir et gérer plusieurs Projets, reliés entre eux par l'association "projets". L'Application-Cadre (Projet) est responsable de cette association.

4.2.2 Application-Cadre (FA)

4.2.2.1 Généralités

Une Application-Cadre est capable de gérer tout type d'équipement en intégrant les DTM correspondants sans qu'une connaissance spécifique des dispositifs ou des bus de terrain soit nécessaire. En fonction de l'utilisation prévue, les Applications-Cadres peuvent présenter des aspects et des caractéristiques différents (par exemple, système de configuration ou d'ingénierie autonome d'un DCS). L'Application-Cadre comprend généralement des applications client axées sur des aspects spécifiques tels que configuration, observation, planification E/S, et qui utilisent le service fourni par les DTM.

L'Application-Cadre constitue l'environnement d'exécution pour les DTM. Elle fournit les services décrits en 7.7 que les DTM hébergés peuvent utiliser.

4.2.2.2 Communication système

Si l'Application-Cadre comporte des capacités de communication intégrées, elle est alors capable de fournir des Voies de Communication (voir 4.2.4.3) qui assurent les services d'accès au bus de terrain. Dans ce cas, elle contrôle totalement la communication et est capable d'effectuer des actions de bas niveau telles que la surveillance ou le filtrage. Les Applications-Cadres sans capacités de communication intégrées utilisent des DTM de Communication (voir 4.2.3.2.2). La Figure 3 représente la relation entre l'Application-Cadre et l'objet Voie de Communication.

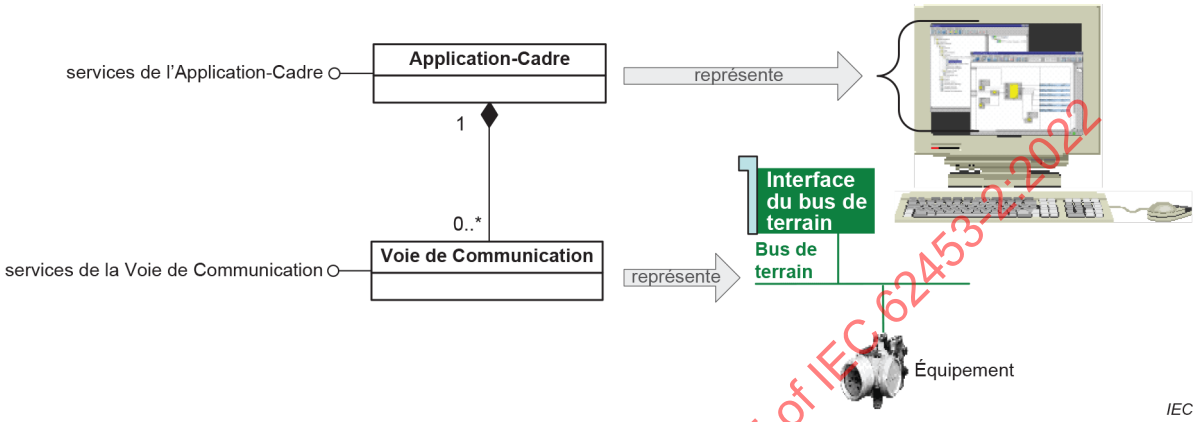


Figure 3 – Application-Cadre avec Voie de Communication intégrée

Les objets Voie de l'Application-Cadre constituent la passerelle entre la communication spécifique aux FDT et celle spécifique à l'Application-Cadre. Du côté de l'Application-Cadre, la Voie de Communication peut être une carte de circuit imprimé E/S ou une topologie d'ingénierie comportant des unités de traitement et des systèmes de bus propriétaires.

4.2.3 Gestionnaire de Type d'Équipement (DTM)

4.2.3.1 Vue d'ensemble de la logique métier (Business Logic) du DTM

La logique métier du DTM (DTM Business Logic - DTM BL) encapsule des fonctions spécifiques à un équipement et des définitions spécifiques à un protocole. Ainsi, une Application-Cadre est capable de gérer tout type d'équipement en intégrant la logique métier du DTM correspondante sans qu'une connaissance spécifique des dispositifs ou des bus de terrain soit nécessaire.

La DTM BL fournit les services décrits en 7.2 (voir la Figure 4). D'un point de vue de l'Application-Cadre, ces services proposent toutes les interactions relatives à la gestion et aux tâches avec la fonctionnalité de l'équipement.

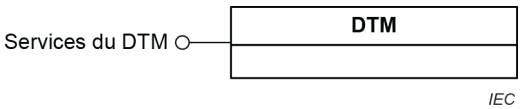


Figure 4 – Gestionnaire de Type d'Équipement (DTM)

Un DTM peut constituer la structure de l'équipement avec le type de données net:DtmDeviceInstanceTopology (voir l'Article A.12). Cette information peut être extraite par le service GetActiveTypeInfo (voir 7.2.3.4). La structure de l'équipement peut être décrite par une liste de net:Module. Chaque module peut comporter un certain nombre de propriétés, y compris les données de configuration, les Voies de Processus et les Voies de Communication.

Un DTM présente une liste des types d'équipements pris en charge avec le type de données fdt:DtmDeviceTypes (voir 4.8.1).

4.2.3.2 Catégories de DTM

4.2.3.2.1 Généralités

Les DTM représentent des équipements de mesure et de commande, des modules, des passerelles ou des blocs, ainsi que des interfaces de communication. Les DTM qui représentent ces équipements ne diffèrent que par la disponibilité des données, des fonctions et des interfaces utilisateur fournies. Les différentes catégories de DTM correspondent aux entités physiques, qui sont représentées par un DTM. Les catégories définies de DTM ne peuvent pas décrire des règles pour tous les équipements physiques existants, mais il est prévu qu'elles fournissent une ligne directrice générale pour les catégories communes d'équipements physiques.

4.2.3.2.2 DTM de Communication

Un DTM de Communication est une catégorie particulière de DTM qui comprend le logiciel d'application pour un matériel de communication spécifique. Par exemple, il peut prendre en charge les réglages de la configuration tels que le débit en bauds, les bits de départ et d'arrêt, etc.

Un DTM de Communication doit fournir des Voies de Communication (voir 4.2.4.3) qui assurent les services d'accès au bus de terrain (voir la Figure 5). Le nombre de voies fournies peut dépendre de la configuration du DTM de Communication (c'est-à-dire qu'un DTM peut ne fournir temporairement aucune voie, par exemple pendant la reconfiguration).

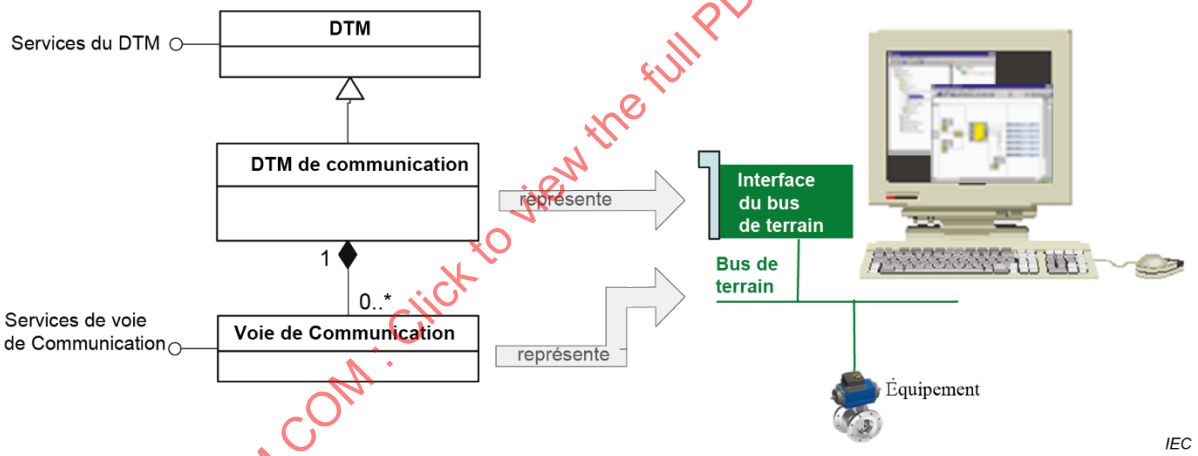


Figure 5 – DTM de Communication

Les Voies de Communication fournies par un DTM par rapport aux voies fournies par une Application-Cadre (voir 4.2.2) présentent l'avantage de pouvoir être utilisées par des Applications-Cadres différentes. Chaque Application-Cadre peut ainsi étendre le nombre de protocoles auxquels le système est capable d'accéder. Les deux manières de fournir des Voies de Communication peuvent coexister dans une Application-Cadre.

4.2.3.2.3 DTM d'Équipement

Un DTM d'Équipement est une catégorie particulière de DTM qui comprend le logiciel d'application pour un dispositif de terrain type, par exemple un transmetteur de pression ou une valve. Un DTM de ce type, par exemple, prend en charge des fonctions de configuration et de paramétrage de l'équipement (voir la Figure 6).

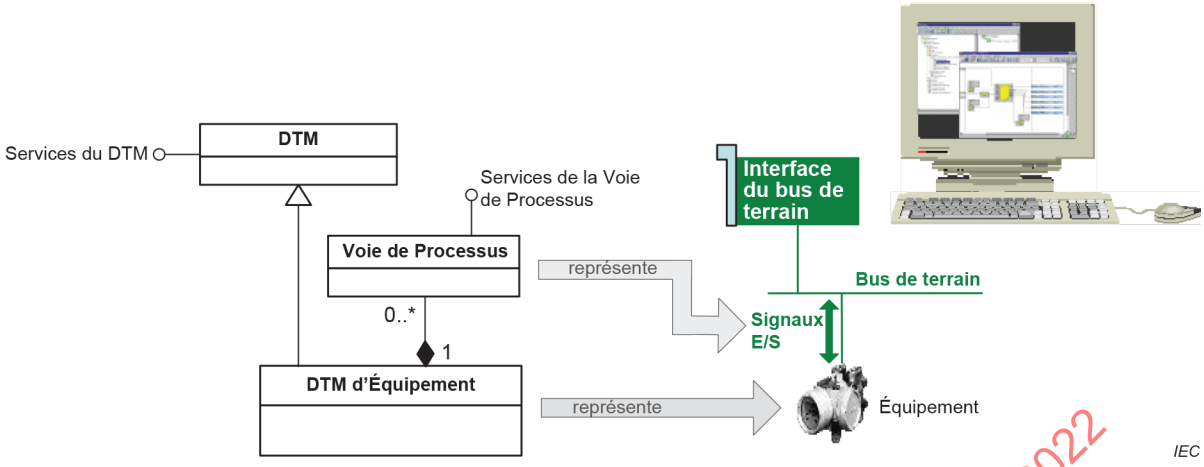


Figure 6 – DTM d'Équipement

Lorsque le système hôte doit pouvoir avoir accès aux signaux E/S de l'équipement ou aux valeurs de processus, le DTM d'Équipement doit alors fournir des objets Voies de Processus (voir 4.2.4.2).

4.2.3.2.4 DTM de Passerelle

Un DTM de Passerelle est une catégorie particulière de DTM qui comprend le logiciel d'application pour un équipement qui relie différents segments de bus de terrain (voir la Figure 7).

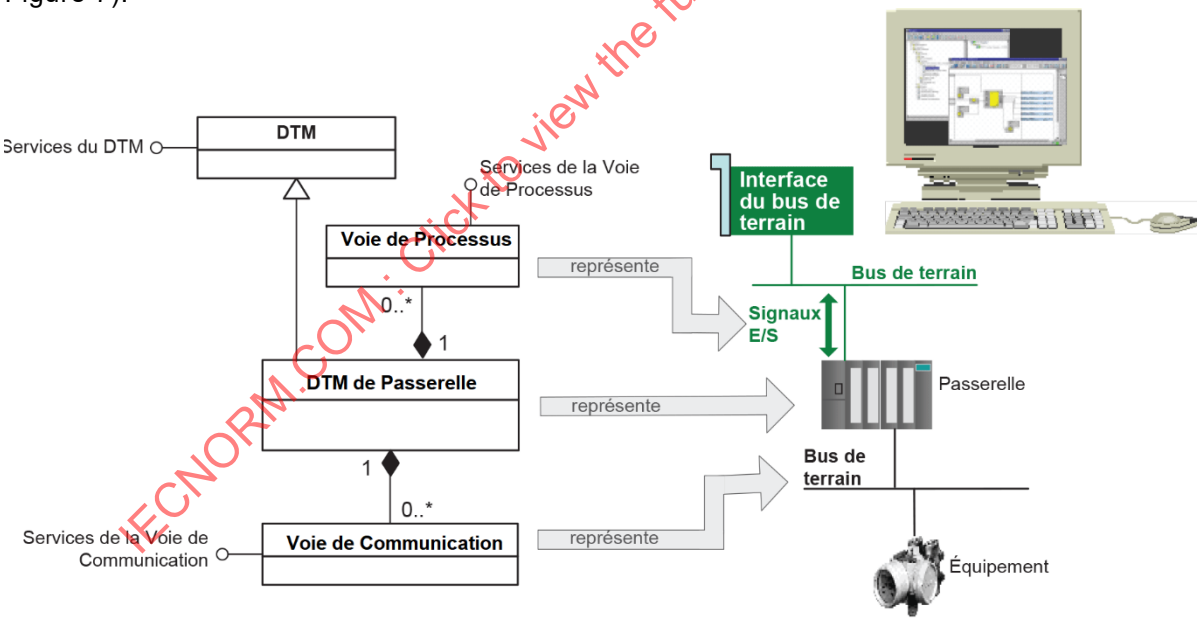


Figure 7 – DTM de Passerelle

Ce type de DTM peut être considéré comme une combinaison d'un DTM de Communication et d'un DTM d'Équipement. Il doit fournir des Voies de Communication (voir 4.2.4.3) afin d'assurer l'accès au bus de terrain relié et aux Voies de Processus (voir 4.2.4.2) du bus de terrain auquel il est relié, lorsqu'il est exigé que le système hôte ait accès aux signaux E/S ou aux valeurs de processus.

Il peut exister une relation entre les signaux E/S fournis par un équipement de passerelle et l'interface de communication de cet équipement (par exemple, la passerelle fournit des signaux E/S en provenance d'un équipement relié à l'interface de communication). Dans ce type de cas, la relation est représentée comme une relation entre les Voies de Processus et la Voie de Communication.

Un DTM Enfant est relié au DTM de Passerelle par l'intermédiaire de sa Voie de Communication. L'Application-Cadre est chargée d'établir et de gérer la liaison entre les DTM. Le DTM d'Équipement est capable de communiquer avec son équipement par l'intermédiaire des services fournis par la Voie de Communication. Ce concept, appelé “Communication Imbriquée”, est détaillé en 4.7.

4.2.3.2.5 DTM d'Équipement composite

Certains équipements sont composés de modules et de sous-modules. Les modules matériels sont branchés à des emplacements physiques de l'équipement composite (équipement modulaire). Par exemple, un module E/S peut être branché à un équipement E/S à distance. La communication entre l'équipement composite et les modules est habituellement assurée par un bus de fond de panier. La fonction du bus de fond de panier est définie par le fabricant et elle est souvent fondée sur des définitions de protocole privées. Un équipement composite de ce type peut être représenté par une hiérarchie de DTM telle qu'elle est représentée à la Figure 8.

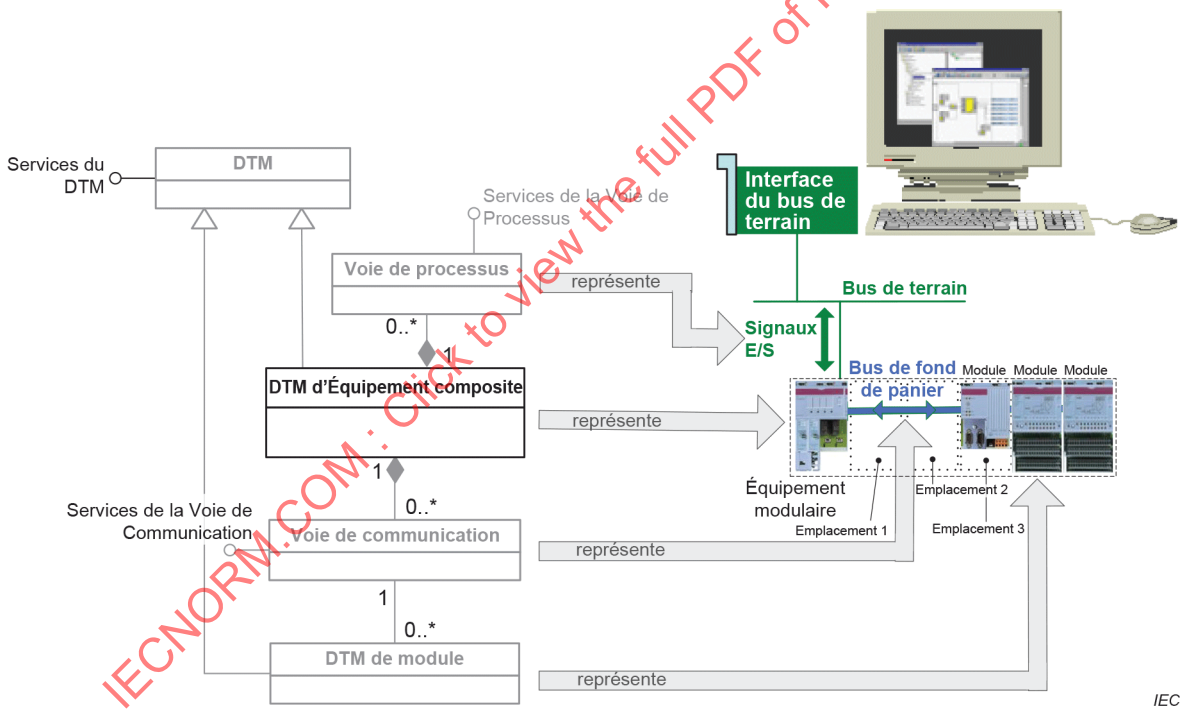


Figure 8 – DTM d'Équipement composite

Un DTM d'Équipement Composite constitue l'élément racine de la représentation complète d'un équipement. Ce DTM doit fournir des Voies de Communication (voir 4.2.4.3) qui représentent le bus de fond de panier. Selon la conception du logiciel du DTM d'Équipement Composite, une voie unique peut être fournie pour représenter le bus de fond de panier dans son ensemble. Plusieurs voies peuvent également être fournies pour représenter les emplacements auxquels sont branchés les modules.

Si l'équipement composite fournit des données de processus, le DTM d'Équipement Composite doit alors fournir les Voies de Processus (voir 4.2.4.2). Si l'équipement composite transmet des données de processus générées dans les modules d'équipement, les Voies de Processus fournies par un DTM d'Équipement composite peuvent alors être constituées de Voies de Processus fournies par les DTM de Module.

4.2.3.2.6 DTM de Module

Un DTM de Module est une catégorie particulière de DTM qui comprend le logiciel d'application pour un module matériel ou logiciel d'un équipement (voir la Figure 9).

Certains équipements sont subdivisés en modules et sous-modules. Un module est soit un module matériel, soit un module logiciel. Les modules matériels sont branchés à des emplacements physiques de l'équipement. Par exemple, un module E/S peut être branché à un équipement E/S à distance. Un équipement modulaire de ce type peut être représenté par plusieurs DTM comme cela est représenté à la Figure 9.

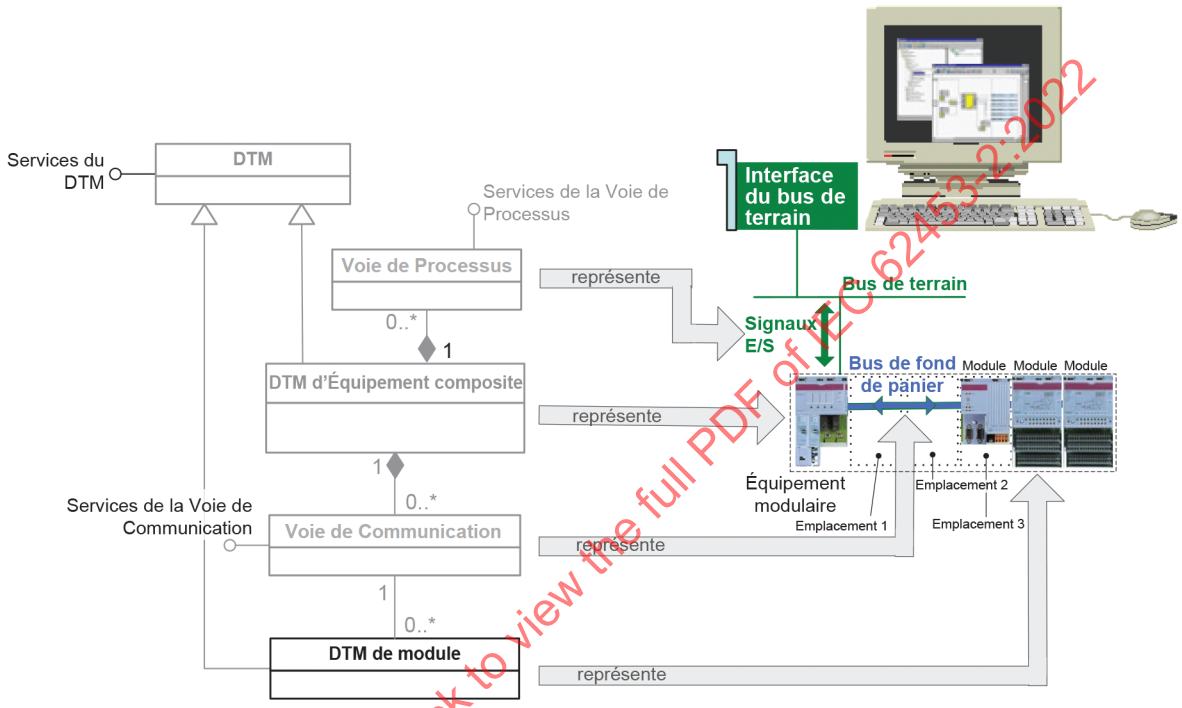


Figure 9 – DTM de Module

Les modules sont représentés par les DTM de Module. Le DTM d'Équipement Composite et les DTM de Module sont généralement expédiés ensemble et fournis par le fournisseur de l'équipement modulaire. La relation entre le DTM d'Équipement Composite et les DTM de Module associés (et leurs DtmTypes respectifs) est définie par un ProtocolId spécifique pour le protocole de bus de fond de panier.

Un DTM Module est relié au DTM d'Équipement Composite par la Voie de Communication du DTM d'Équipement Composite. L'Application-Cadre est chargée d'établir et de gérer la liaison entre les DTM. Le DTM Module est capable de communiquer avec son module par l'intermédiaire des services fournis par la Voie de Communication. Cela signifie que le concept de “Communication Imbriquée” (voir 4.7) s'applique aussi aux DTM d'Équipement Composite et aux DTM Modules associés.

Si les modules d'équipement fournissent des données de processus, il convient alors que les DTM de Module respectifs fournissent les Voies de Processus.

Si les modules d'équipement fournissent l'accès à la communication subordonnée (par exemple, dans une passerelle modulaire), le DTM de Module respectif doit alors fournir ses propres Voies de Communication.

4.2.3.2.7 Gestionnaire de Type de Blocs (BTM)

Un BTM est une catégorie particulière de DTM qui comprend le logiciel d'application pour les objets accessibles à l'intérieur d'un équipement. Ces objets logiciels sont souvent définis comme des blocs, tels que des blocs de ressources, blocs de transducteurs et blocs fonctionnels (voir la Figure 10).

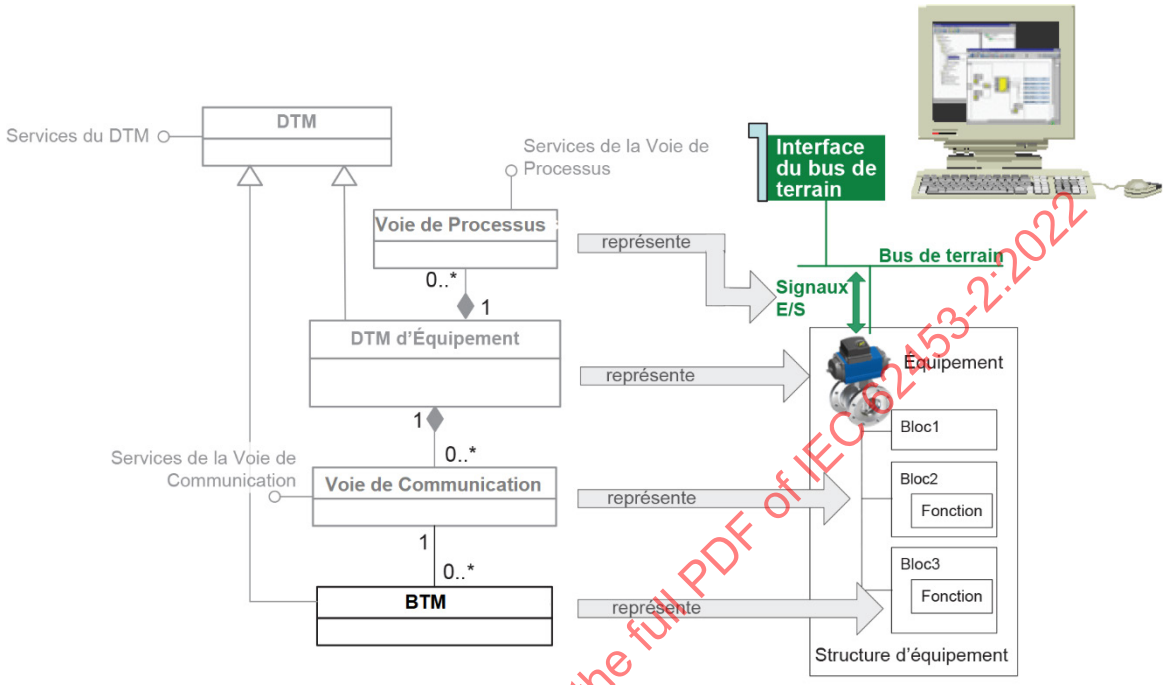


Figure 10 – Gestionnaire de Type de Blocs (BTM)

Un DTM d'Équipement constitue l'élément racine de la représentation complète d'un équipement. Ce DTM d'Équipement doit fournir des Voies de Communication (voir 4.2.4.3) qui assurent des services d'accès aux informations des blocs au sein de l'équipement.

Un équipement peut héberger des blocs normalisés, des blocs spécifiques à un fournisseur et des blocs spécifiques à un équipement. Tous les types de blocs sont représentés par les BTM. Les blocs logiciels sont plus souples que les modules d'équipement (voir 4.2.3.2.6). Par exemple, des blocs peuvent être déplacés entre des équipements (les blocs normalisés peuvent même être déplacés entre des équipements de type différent). Le DTM d'Équipement et les BTM pour les blocs spécifiques à un équipement sont généralement expédiés ensemble et fournis par le fournisseur d'équipements. Les BTM pour les blocs normalisés peuvent être fournis séparément et peuvent être utilisés avec tous les DTM pour les équipements capables d'héberger les blocs respectifs.

Un BTM est relié au DTM d'Équipement hôte par sa Voie de Communication, indiquant que les blocs respectifs sont hébergés dans l'équipement physique. Le BTM est capable d'interagir avec son bloc par le biais des services fournis par la Voie de Communication du DTM d'Équipement. La Voie de Communication représente une communication interne sur l'équipement; elle permet principalement de gérer la liaison entre le BTM et le bloc. Les protocoles pris en charge par la Voie de Communication sont utilisés pour gérer les BTM qui peuvent être reliés à un DTM (des protocoles normalisés pour les blocs normalisés, des protocoles spécifiques pour les blocs spécifiques). La Voie de Communication peut appliquer des règles de validation supplémentaires. L'Application-Cadre est chargée d'établir et de gérer la liaison entre le DTM d'Équipement et les BTM.

Le concept des BTM est indépendant du protocole du bus de terrain. Cependant, un protocole peut exiger la modélisation des structures des équipements en blocs. Par conséquent, l'utilisation et la manière d'utiliser le concept des BTM afin de modéliser les structures de l'équipement sont définies dans les documents IEC 62453-3xy qui décrivent l'intégration des profils de protocole dans les FDT.

Si les blocs fournissent des données de processus, les BTM respectifs peuvent fournir les Voies de Processus.

4.2.3.3 Fonctions du DTM

4.2.3.3.1 Objet Présentation

Chacun des objets spécifiques à l'équipement (DTM, BTM et Voie de Communication) peut être paqueté avec les objets Présentation qui prennent en charge les services définis en 7.3 (voir la Figure 11).

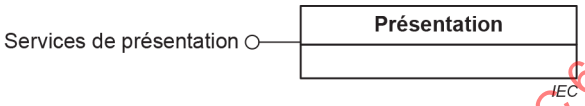


Figure 11 – Objet Présentation

Ces Objets Présentation sont des objets spécifiques à la mise en œuvre qui fournissent une interface utilisateur graphique pour l'objet métier.

L'Objet Présentation peut être un type d'objets, qui permet l'intégration de l'Application-Cadre à l'interface utilisateur graphique. Il peut également s'agir d'un objet qui fonctionne à l'extérieur de l'interface utilisateur graphique de l'Application-Cadre (par exemple, un programme exécutable autonome).

4.2.3.3.2 Fonctions de commande (Command)

Les fonctions de commande sont utilisées pour exécuter des actions (commandes) sans interface utilisateur sur le DTM. Ces fonctions sont exécutées avec le service InvokeFunction (voir 7.2.5.2).

Un exemple de fonctions de commande est: Reset device (Réinitialiser l'équipement).

4.2.3.3.3 Fonction Statique (Static)

La fonction statique est un concept qui permet de traiter les informations issues d'un équipement sans exécuter le DTM.

Pour l'observation et la surveillance d'une installation, il est souvent nécessaire de récupérer quelques données de nombreux équipements. Dans ces cas d'utilisation, l'effort pour démarrer et gérer de nombreux DTM peut s'avérer trop élevé. D'autre part, dans ces cas d'utilisation, il est possible que la lecture d'un seul élément de l'équipement ne suffise pas. Parfois, il est nécessaire de traiter ou de combiner les données lues dans l'équipement avec des informations complémentaires afin de calculer les informations correctes.

Une Fonction Statique fait partie intégrante d'un paquetage de DTM. Le DTM fournit des informations relatives à la Fonction Statique. Une Fonction Statique ne comporte aucun accès aux données d'un DTM ni aucune relation avec celles-ci.

Les Fonctions Statiques sont exécutées en utilisant un objet StaticFunction Provider. L'Application-Cadre est chargée de créer les objets StaticFunctionProvider. Un seul objet StaticFunctionProvider doit être présent par équipement. L'objet est initialisé par les données de communication nécessaires. Sur un seul et même objet StaticFunctionProvider, l'exécution d'une Fonction Statique peut être lancée plusieurs fois en parallèle. Il incombe à StaticFunctionProvider de traiter successivement les fonctions selon leur ordre de lancement.

Il est recommandé qu'un DTM fournisse des Fonctions Statiques pour prendre en charge des cas d'utilisation tels qu'ils sont définis par les spécifications relatives à l'intégration des profils de protocole dans les FDT.

NOTE Il n'est pas attendu que des Fonctions Statiques fournissent une fonctionnalité spécifique pour une Application-Cadre. Au contraire, il est attendu que les groupes de travail sur des profils (par exemple, un groupe de travail qui définit un profil pour la prise en charge de fonctions MES ou un groupe de travail qui définit un profil de DTM pour la prise en charge d'une maintenance prédictive) définissent des fonctions normalisées qui sont utilisées dans de nombreuses applications.

Exemples de fonctions statiques:

– GetDeviceStatus

  qui fournit le statut actuel de l'équipement au format FDT (voir Fdt.Dtm.DeviceStatus)

– GetProcessValue

  qui fournit la valeur actuelle de processus de l'équipement (entrée ou sortie) dans un format normalisé

– GetOperationTime

  qui fournit la durée de fonctionnement d'un équipement dans un format normalisé

4.2.4 Objet Voie (Channel)

4.2.4.1 Vue d'ensemble des voies

Il existe trois types différents de voies, à savoir: les Voies de Communication, les Voies de Processus et les combinaisons des deux comme cela est représenté à la Figure 12.

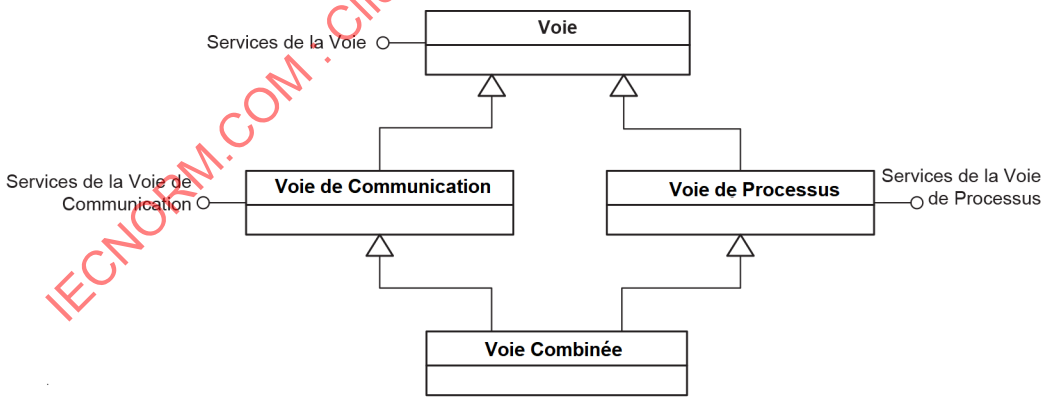


Figure 12 – Objet Voie

4.2.4.2 Voie de Processus

Les signaux d'entrée et de sortie fournis par les équipements sont générés et intégrés à la planification des fonctions du système de commande. Si l'équipement fournit des signaux E/S, le DTM associé doit proposer des informations sur les signaux E/S de son équipement. Ces informations sont acheminées par les Voies de Processus.

Une Voie de Processus prend en charge les services définis en 7.5. Ces services fournissent, par exemple, un accès aux informations relatives aux signaux E/S telles que l'adressage, le type de signal et de données, mais pas à la valeur actuelle du signal E/S proprement dit.

La mise en œuvre de la Voie de Processus et de ses services respectifs dépend de la technologie de mise en œuvre. Elle est différente pour l'IEC TR 62453-41 et l'IEC TR 62453-42.

Les informations relatives aux signaux E/S fournies par la Voie de Processus sont spécifiques à un protocole. Les informations que doivent fournir les Voies de Processus sont définies dans le document IEC 62453-3xy approprié qui décrit l'intégration des profils de protocole dans les FDT.

Le nombre et le type de signaux E/S peuvent dépendre de la configuration de l'équipement. Par conséquent, le nombre de Voies de Processus et les informations fournies peuvent également dépendre de la configuration de l'équipement effectuée par l'utilisateur. Une Application-Cadre est capable d'informer le DTM, en positionnant le paramètre ProtectedByChannelAssignent par le biais du service WriteChannelData (voir 7.5.2), que des modifications ultérieures doivent être interdites. L'intégration effective de la voie à la planification fonctionnelle du système de commande (La Voie Hôte est attribuée) justifie cette interdiction. Dans ce cas, le DTM ne doit accepter aucune modification relative à cette voie.

4.2.4.3 Voie de Communication

Une Voie de Communication fournit l'accès au bus de terrain et elle est dépendante de la technologie de communication. Elle représente le point d'entrée d'un bus de terrain, d'un bus de fond de panier spécifique à un fournisseur ou d'une liaison point à point. Une Voie de Communication peut également représenter une communication logique avec des blocs au sein d'un équipement. Une Voie de Communication peut être fournie par une Application-Cadre (voir 4.2.2) ou par un DTM (voir 4.2.3).

Une Voie de Communication fournit des services de communication indépendants du matériel de communication et du protocole de bus de terrain. En général, les services spécifiques à un protocole sont mis en correspondance biunivoque avec les services fournis par la voie. Les interfaces fournissent des méthodes pour

– balayer le bus de terrain pour trouver les équipements disponibles,
– se connecter à un équipement,
– envoyer des messages de communication, et
– se débrancher d'un équipement.

Les services utilisent des arguments de base qui sont étendus pour une communication spécifique à un protocole par les spécifications IEC 62453-3xy (voir la Figure 13).

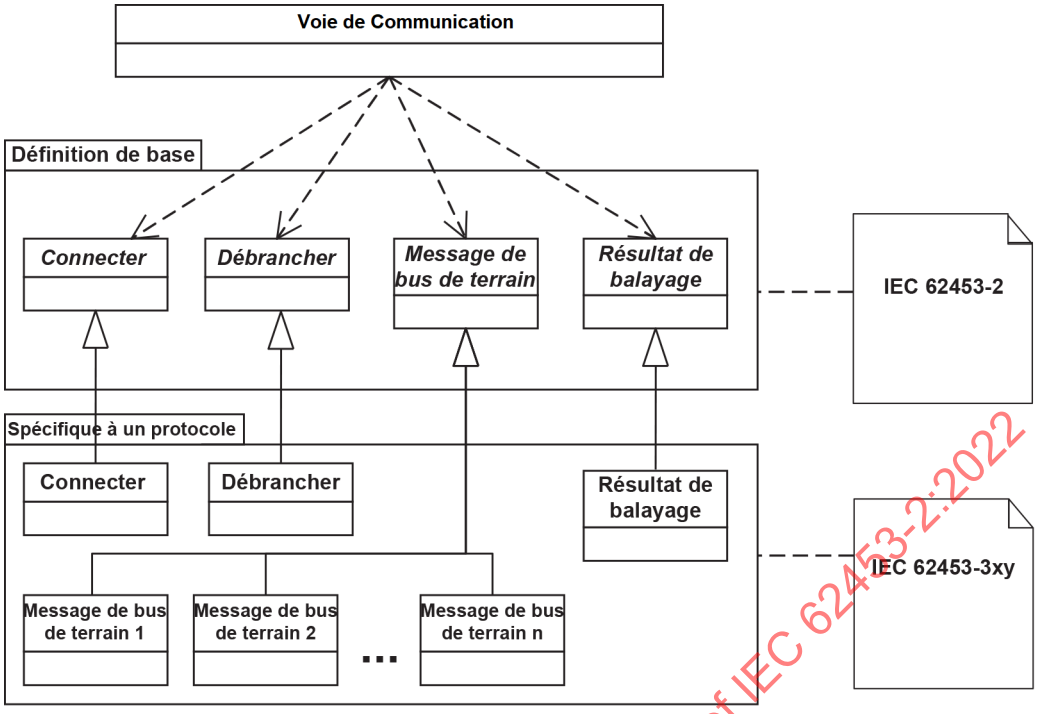


Figure 13 – Voie de Communication

Les FDT permettent également aux fabricants de définir la prise en charge pour des protocoles de bus propriétaires, qui ne sont pas partie intégrante de la norme, par exemple pour une communication de bus de terrain ou point à point spécifique au fabricant.

Les services fournis par des Voies de Communication peuvent être utilisés par l'Application-Cadre ou par des DTM pour échanger des données avec un équipement connecté ou pour initier une fonction (par exemple, demande d'identification, réinitialisation de l'équipement, diffusion, etc.). Le concept général de communication est détaillé en 4.7.

Une Voie de Communication doit être capable de gérer plusieurs connexions à la fois, si le protocole de communication prend en charge cette fonction. Cette disposition signifie qu'une Voie de Communication doit être capable de prendre en charge aussi bien plusieurs DTM qui établissent des connexions à différents équipements que plusieurs DTM qui établissent des connexions au même équipement. Selon le protocole pris en charge, il peut également être possible qu'un seul et même DTM établisse plusieurs connexions à un seul et même équipement. La Voie de Communication doit garantir que la liaison à un équipement est établie comme une connexion entre homologues. Les services qui doivent être fournis par une Voie de Communication sont définis en 7.6. Les services d'origine relatifs à la communication (voir 7.6.1) définissent uniquement le cadre pour l'interaction avec un équipement connecté. Les données (paramètres) qui doivent être acceptées ou retournées par le service sont définies dans les documents IEC 62453-3xy qui décrivent l'intégration des profils de protocole dans les FDT.

Dans le cas de protocoles spécifiques à un fournisseur (par exemple, bus de fond de panier ou point à point), des services spécifiques à un fournisseur peuvent se substituer aux services d'origine relatifs à la communication (voir 7.6.1). En suivant cette approche, seuls les DTM fournis par ce fournisseur sont capables de communiquer entre eux. Par conséquent, une catégorie de bus spécifique au fournisseur (voir 4.3) doit être définie. Pour les protocoles de bus de terrain indépendants du fournisseur, un document IEC 62453-3xy qui décrit l'intégration des profils de protocole dans les FDT doit être fourni et les services normalisés de communication doivent être utilisés.

4.2.4.4 Voie combinée de Processus et de Communication

Une voie combinée de Processus et de Communication appartient généralement à un DTM de Passerelle (voir 4.2.3.2.4). Elle représente le signal E/S présenté par le biais du bus de terrain auquel l'équipement de passerelle est connecté, ce qui correspond à la communication cyclique sur le bus de terrain relié. Ce type de voie représente également le point d'entrée du bus de terrain relié (ou liaison point à point) comme cela est représenté à la Figure 14.

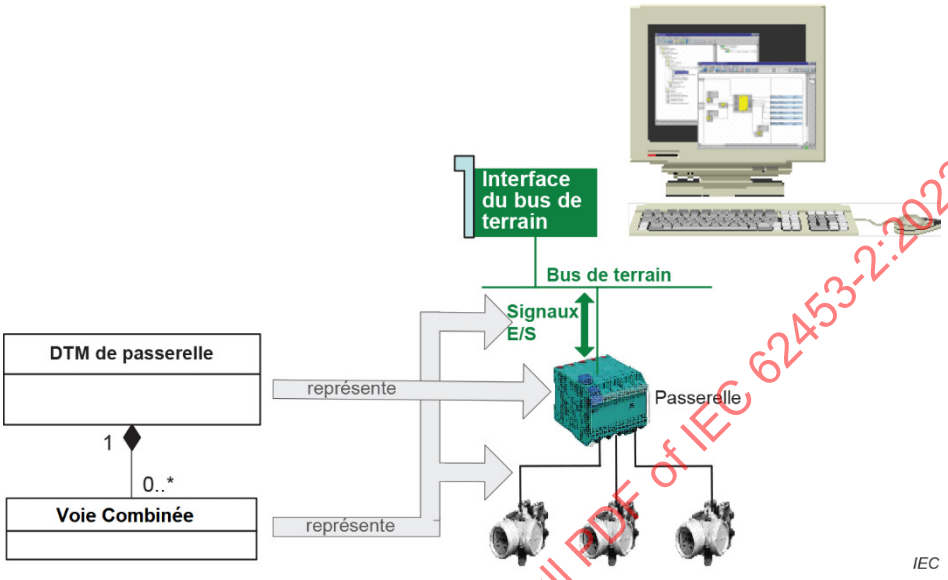


Figure 14 – Voie combinée de processus/communication

En d'autres termes, ce type de voie doit prendre en charge les services de la Voie de Processus (voir 7.5) pour le bus de terrain auquel la passerelle est connectée et les services de la Voie de Communication (voir 7.6) pour accéder aux équipements connectés au bus de terrain relié (ou liaison point à point).

4.3 Modularité

Des protocoles de bus de terrain différents utilisent des modèles d'équipement différents. Les FDT prennent en charge les différentes approches suivantes pour décrire la structure de l'équipement:

- DTM monolithique avec DtmDeviceInstanceTopology,
- DTM de Module, et
- BTM.

4.4 Catégories de bus

Un DTM présente les informations relatives aux définitions spécifiques à un protocole encapsulé. Les informations retournées par le service GetTypeInformation (voir 7.2.3.1) contiennent lesdites catégories de bus qui sont des Identificateurs Uniques universels pour un protocole de bus de terrain (ou une communication point à point).

Le concept relatif aux FDT fait une distinction entre les catégories de bus prises en charge et celles qui sont exigées:

- les catégories de bus prises en charge font référence aux services de communication spécifiques à un protocole fournis par un DTM (Voies de Communication correspondantes) pour accéder à un bus de terrain;
- les catégories de bus exigées font référence aux services de communication spécifiques à un protocole exigés par un DTM pour échanger des données avec son équipement.

4.5 Identification

4.5.1 Identification d'instance de DTM

4.5.1.1 Marqueur système

Une Application-Cadre FDT doit assigner un identificateur unique pour chaque instance de DTM. Cet identificateur unique est appelé "Marqueur système" ("System Tag"). Le Marqueur système est utilisé par les DTM

– pour la navigation dans la topologie des FDT,

– pour la gestion des DTM Enfants dans la topologie des FDT (par exemple, réglage d'adresse).

4.5.1.2 Étiquette de l'interface utilisateur graphique système

L'étiquette DtmSystemGuiLabel est un identificateur interprétable par l'utilisateur qui permet d'identifier une instance de DTM dans le Projet et l'interface utilisateur graphique de DTM associée dans l'affichage à l'attention de l'utilisateur. Il convient que l'Application-Cadre et le DTM utilisent la DtmSystemGuiLabel afin d'identifier des dialogues et des fenêtres appartenant à l'instance de DTM.

L'étiquette DtmSystemGuiLabel peut représenter un identificateur de l'équipement (par exemple, le marqueur d'équipement) fourni dans l'étiquette NetworkDataInfo.Label. Un DTM peut utiliser une étiquette NetworkDataInfo.Label afin de fournir une valeur initiale pour l'étiquette DtmSystemGuiLabel. Il est recommandé que l'Application-Cadre utilise la valeur de NetworkDataInfo.Label lors de la réalisation de DtmSystemGuiLabel.

4.5.1.3 Identification de matériel

Un DTM prend en charge le service HardwareInformation (voir 7.2.3.3) qui permet de lire des informations relatives à l'équipement en ligne à partir de l'équipement connecté (voir la Figure 15).

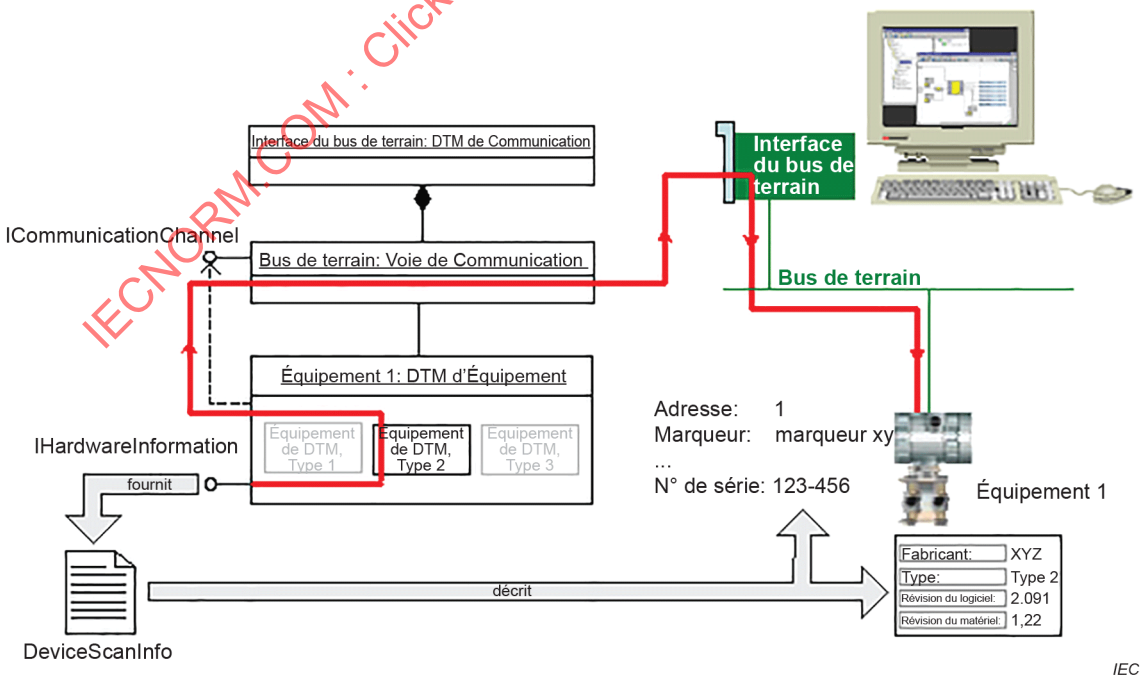


Figure 15 – Identification d'équipements connectés

Le service retourne les informations relatives au type d'équipement telles que le fabricant, le type, la version du matériel et du logiciel intégré, ainsi que les informations relatives aux instances de l'équipement telles que l'adresse, le marqueur et le numéro de série du bus de terrain. Ces informations sont également spécifiques au bus de terrain tout comme les informations retournées par le service GetIdentificationInformation. Par conséquent, le format concret spécifique à un protocole et la conversion dans un format indépendant du protocole qui peuvent être utilisés par l'Application-Cadre sans connaissance spécifique du bus de terrain sont définis par le document IEC 62453-3xy qui décrit l'intégration des profils de protocole dans les FDT.

4.6 Topologie du système et des FDT

L'Application-Cadre est chargée de la gestion de la topologie du système. Cette disposition signifie que l'Application-Cadre doit organiser le cheminement pour accéder à un équipement du plan. Un DTM ne doit pas avoir besoin de gérer une quelconque information relative à la topologie du système.

La gestion de la topologie du système est spécifique à l'Application-Cadre. Certaines Applications-Cadres peuvent exiger des interactions avec l'utilisateur, d'autres peuvent prendre en charge des opérations automatiques telles que l'importation de la topologie ou le balayage du bus de terrain (voir 6.2).

Un DTM présente toutes les informations exigées (voir 7.2.3) qui permettent à l'Application-Cadre (et à l'utilisateur) de choisir le DTM approprié pour un équipement, par exemple, le nom, le fournisseur, la version des types d'équipements pris en charge et les propriétés d'identification correspondantes.

L'Application-Cadre initialise et relie les DTM conformément à la topologie du système. La liaison entre les DTM est dénommée topologie des FDT. La Figure 16 représente la topologie des FDT pour une topologie de système simple avec un bus de terrain et deux équipements connectés.

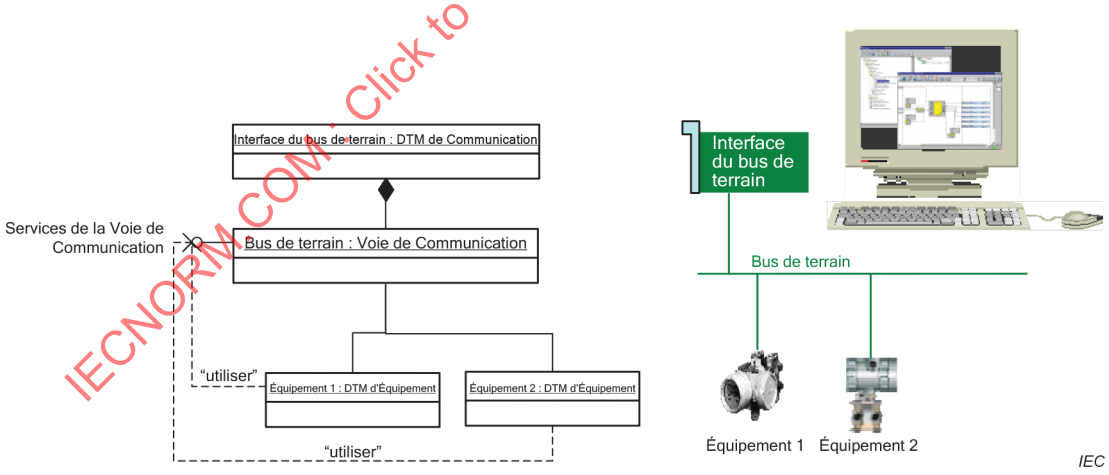


Figure 16 – Topologie des FDT pour une topologie de système simple

Une Voie de Communication est toujours utilisée comme l'élément de liaison entre des DTM. À la Figure 16, les deux DTM d'Équipement sont reliés à la Voie de Communication qui fournit un accès au bus de terrain.

La liaison entre une Voie de Communication et un DTM est établie par l'Application-Cadre. Cependant, la Voie de Communication doit décider (décision finale) si un DTM doit être relié ou non. L'Application-Cadre doit appeler le service ValidateAddChild (voir 7.6.2.1) avant d'établir la liaison. La Voie de Communication doit au moins vérifier si les catégories de bus exigées du DTM à relier sont conformes ou non à ses propres catégories de bus prises en charge. Si tel n'est pas le cas, la liaison doit alors être rejetée. De plus, la Voie de Communication peut effectuer des vérifications supplémentaires, par exemple si le nombre de DTM reliés dépasse une limite donnée. Les diagrammes de séquences en 8.1 détaillent cette séquence.

Ni la Voie de Communication (ou le DTM correspondant) ni le DTM relié ne doivent avoir besoin de gérer les informations relatives à la topologie. L'Application-Cadre prend en charge les demandes d'informations relatives à la topologie par le service GetParentNodes (voir 7.7.3.5) et le service GetChildNodes (voir 7.7.3.6).

La gestion de la topologie des FDT pour une topologie de système plus complexe avec des passerelles, des équipements modulaires ainsi que des équipements avec des blocs, fonctionne exactement de la même manière.

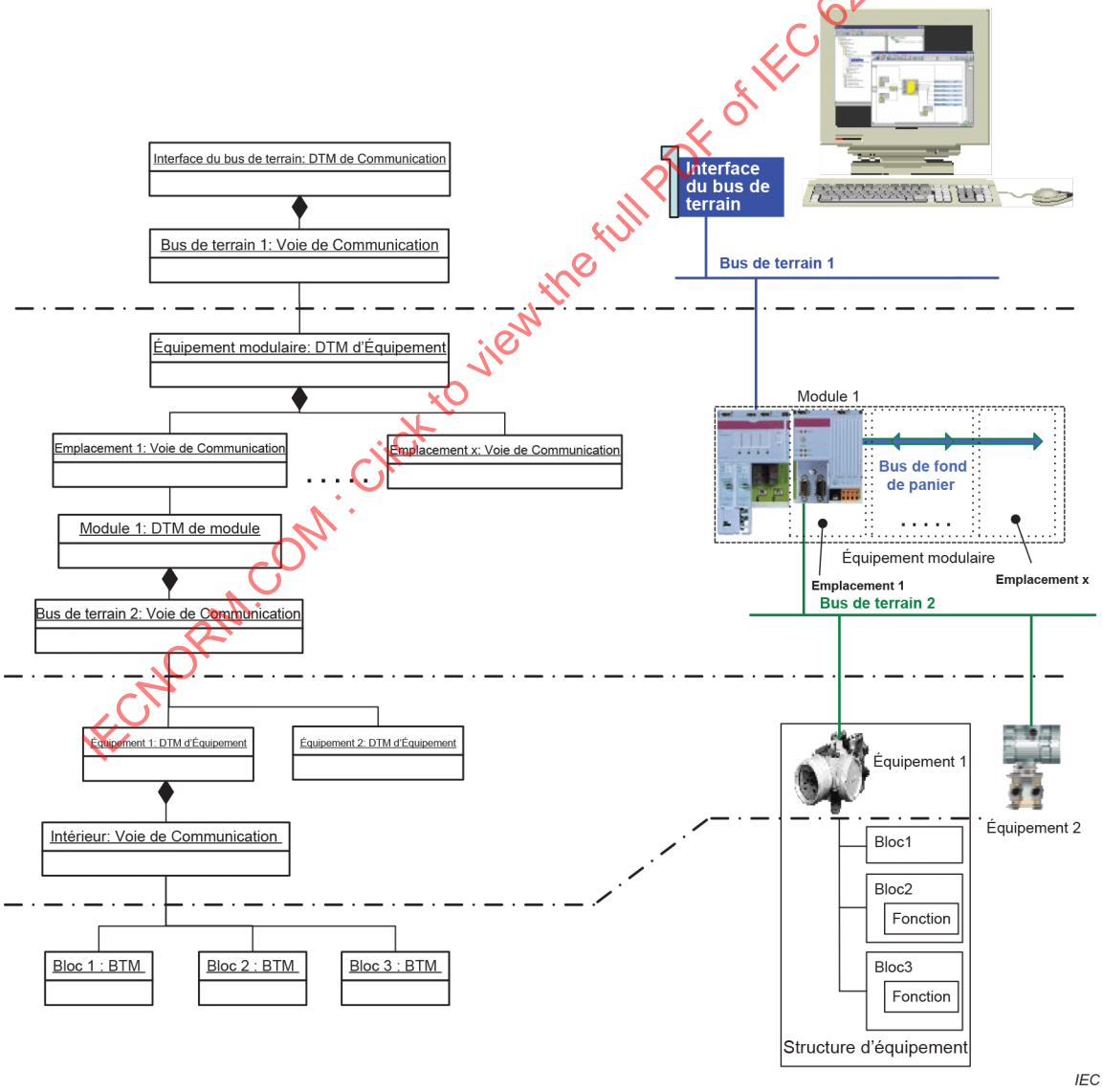


Figure 17 – Topologie des FDT pour une topologie de système complexe

En partant de la racine, tous les DTM sont reliés par le biais de Voies de Communication conformément à la topologie du système physique.

Le DTM de l'élément racine pour un équipement représenté par plusieurs DTM (par exemple, les DTM d'Équipement Modulaire et d'Équipement 1 de la Figure 17) peut prendre en charge la génération automatique de la sous-topologie des FDT qui correspond à la structure complète de l'équipement. L'Application-Cadre prend en charge le service CreateChild (voir 7.7.3.2), DeleteChild (voir 7.7.3.3) et MoveChild (voir 7.7.3.4) pour permettre à un DTM de modifier la topologie des FDT.

4.7 Communication des FDT

4.7.1 Généralités

Dans les FDT, la communication est fondée sur la transmission de demandes de communication du consommateur de communication au fournisseur de communication, ce qui est représenté par une Voie de Communication. La transmission et la réception de demandes de communication sont fondées sur des services asynchrones (voir 8.3).

Afin de présenter des demandes de communication, le consommateur de communication doit d'abord établir une ou plusieurs connexions avec l'équipement physique respectif. Cette connexion permet la gestion de connexions actives par la Voie de Communication.

4.7.2 Traitement des demandes de communication

La Voie de Communication reçoit une ou plusieurs demandes de transaction issues du consommateur de communication et exécute ces demandes. En général, il est attendu que la Voie de Communication traite les demandes de transaction dans l'ordre dans lequel elles sont fournies. Les résultats des demandes de transaction sont renvoyés au client de la Voie de Communication.

La relation entre demandes de communication et réponses de communication peut être gérée par un identificateur (ID). Cette disposition est spécifique à une mise en œuvre. Par exemple, chaque demande de transaction peut être identifiée par un ID qui est le même que celui qui est fourni dans la réponse de transaction.

4.7.3 Traitement des erreurs de communication

Le client de communication doit évaluer chacun des résultats de transaction reçus afin de déterminer les erreurs de communication possibles.

Si un client de communication reçoit des erreurs pour des demandes dont il est à l'origine, il doit alors en informer l'utilisateur. Dans un scénario avec une communication imbriquée, cela signifie que les DTM de Passerelle ne doivent pas donner à l'utilisateur des informations relatives à des erreurs de communication pour des demandes qui proviennent initialement d'un DTM Enfant. Le DTM Enfant est chargé d'informer l'utilisateur.

4.7.4 Traitement des pertes de connexion

Si la connexion à un équipement est irrémédiablement perdue, la Voie de Communication doit mettre fin à la connexion ou aux connexions respectives et envoyer une notification Arrêt prématuré (Abort) au client de communication respectif pour chaque connexion arrêtée (voir 7.6.1.5). AbortMessage fournit la CommunicationReference à la connexion arrêtée. Lorsqu'un DTM crée plusieurs connexions vers l'équipement, chaque connexion est traitée séparément. L'arrêt prématuré de chaque connexion peut dépendre de conditions spécifiques selon le protocole de communication utilisé.

Les déclencheurs spécifiques pour l'envoi de la notification Arrêt prématuré (par exemple, l'équipement ne répond pas) sont définis spécifiquement pour chaque protocole de communication.

Si un DTM de Passerelle reçoit une notification Arrêt prématuré, il doit alors envoyer des notifications Arrêt prématuré à tous les clients de communication connectés.

L'origine de la notification Arrêt prématuré ne doit pas fournir de message utilisateur pour informer l'utilisateur de la perte de connexion. Le client de communication qui reçoit la notification Arrêt prématuré doit informer l'utilisateur de la perte de connexion. Par exemple, si le client de communication comporte des interfaces utilisateur ouvertes, l'utilisateur doit alors être informé de la perte de connexion par une mise à jour du statut de connexion dans l'interface utilisateur. Si le client de communication se compose de plusieurs DTM qui peuvent comporter des connexions ouvertes au même moment (par exemple, un DTM d'Équipement composite avec plusieurs DTM de Module), il convient alors qu'il évite de fournir plusieurs messages utilisateur. Il convient en revanche qu'il informe l'utilisateur de la perte de connexion par un seul message.

Après l'envoi d'une notification Arrêt prématuré, la Voie de Communication ne doit envoyer aucune autre réponse de transaction au client de communication. Il convient que le client de communication ne tienne aucun compte d'une réponse de transaction reçue après la réception d'une notification Arrêt prématuré.

4.7.5 Communication point à point

L'Application-Cadre doit fournir dans chacun des cas une connexion entre homologues entre un DTM et un équipement.

L'Application-Cadre est chargée d'autoriser un DTM à communiquer avec son équipement. L'Application-Cadre doit transmettre la Voie de Communication pour utiliser le DTM en appelant le service SetLinkedCommunicationChannel (voir 7.2.4.2) et permettre la Communication en appelant le service EnabledCommunication avec TRUE (VRAI) (voir 7.2.4.3).

Pour accéder à l'équipement, le DTM utilise les services de la Voie de Communication (voir 7.6.1) comme cela est représenté à la Figure 18.

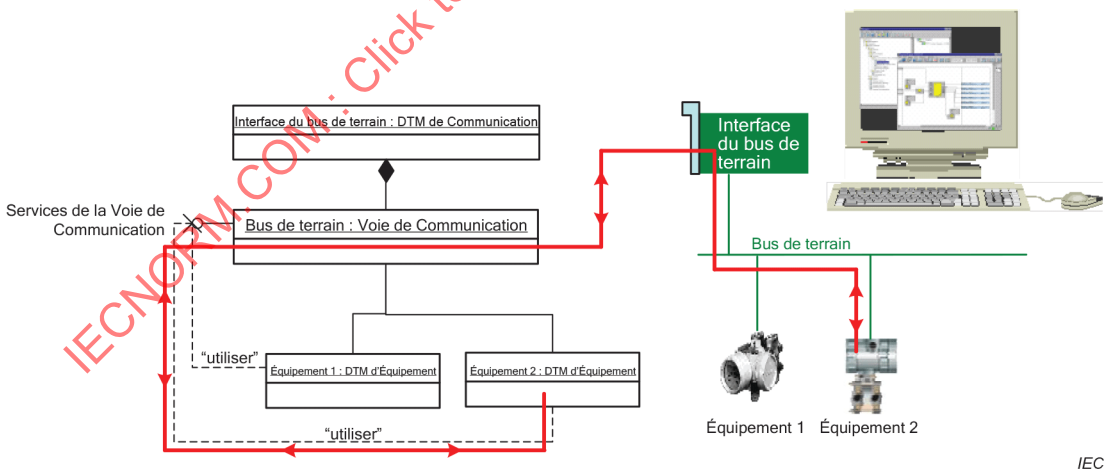


Figure 18 – Communication point à point

Un DTM doit appeler le service Connect (Connexion) de la Voie de Communication (voir 7.6.1.2) pour établir une liaison de communication avec l'équipement. Lorsque la liaison est établie, le DTM est capable de communiquer avec son équipement en appelant le service Transaction (voir 7.6.1.6). Le diagramme en 8.3.2 détaille cette séquence.

Les DTM doivent prendre en considération la disponibilité limitée des ressources de bus de terrain lorsqu'ils se connectent à l'équipement. Cette disposition signifie qu'il convient qu'un DTM établisse seulement autant de connexions que nécessaire et que les connexions doivent être fermées si elles ne sont pas utilisées (par exemple, la fonction en ligne est terminée).

4.7.6 Communication imbriquée

Un DTM ne doit pas avoir besoin de disposer d'informations concernant la topologie du système pour accéder à l'équipement physique correspondant. Par conséquent, le même DTM est capable de communiquer avec les équipements rattachés à différentes interfaces de bus de terrain du même protocole. La communication imbriquée est utilisée lorsque les équipements sont connectés à un sous-système, par exemple si un équipement est connecté à une voie d'un système à distance E/S (voir la Figure 19).

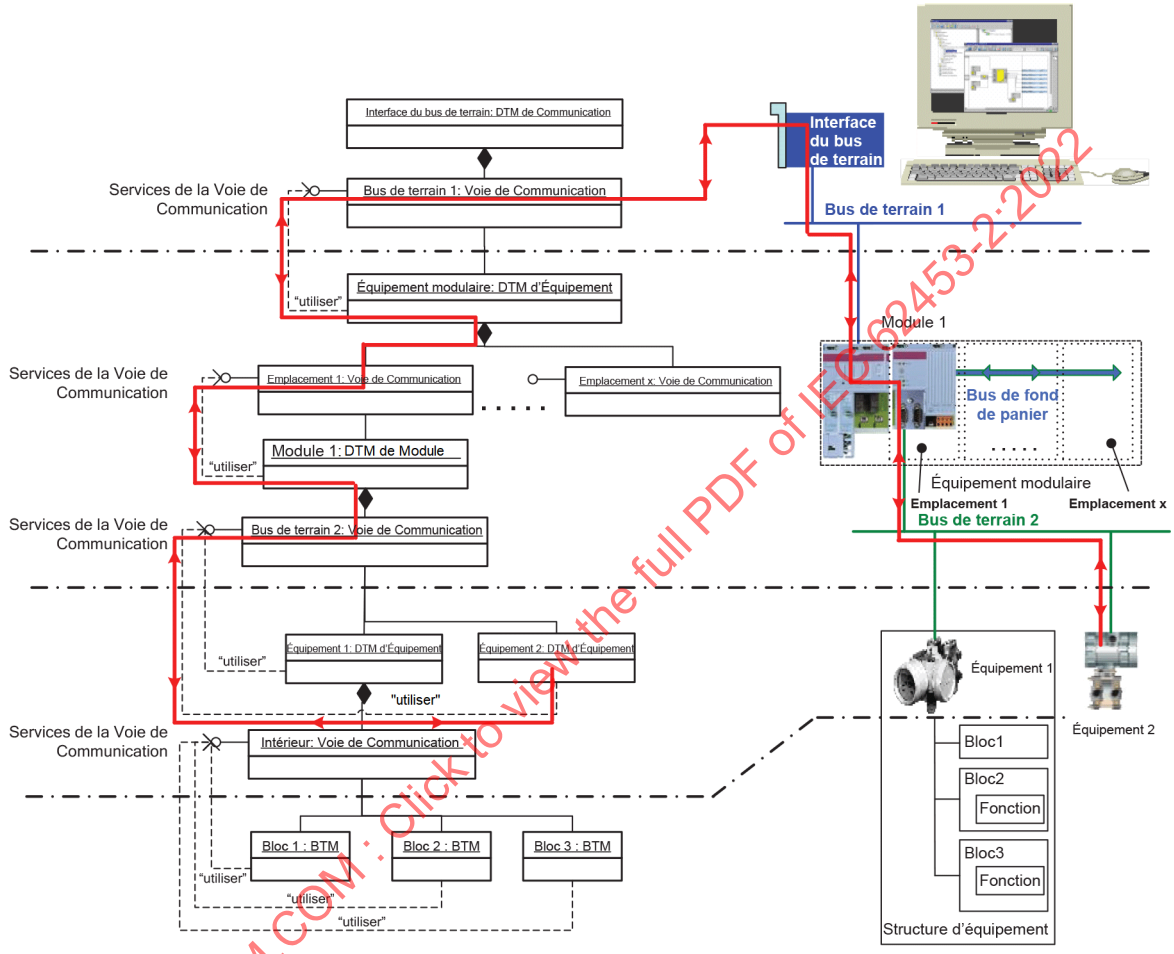


Figure 19 – Communication imbriquée

En partant de la racine, l'Application-Cadre doit gérer la liaison entre les Voies de Communications et les DTM respectifs pour lesquels il convient d'autoriser la communication avec leurs équipements respectifs. À l'instar de la communication point à point, tous les DTM utilisent simplement les services de la Voie de Communication pour communiquer avec leurs équipements sans information de la communication imbriquée (par exemple, le DTM de l'équipement 2 représenté à la Figure 20). Le diagramme en 8.3.3 détaille cette séquence.

La communication avec les FDT suit le concept de communication imbriquée:

- chaque Voie de Communication enveloppe le message de communication provenant du DTM relié situé en dessous, sans en connaître le contenu;
- le DTM relié situé en dessous de la Voie de Communication ne dispose d'aucune information sur la topologie du système;
- le DTM ne doit prendre en charge que les protocoles de communication qui sont pris en charge par l'équipement respectif;
- la communication/le cheminement à travers toute topologie de système est réalisable sans aucune limite.

Se reporter à 8.3.3 pour de plus amples informations sur la séquence de communication relative à la communication imbriquée.

4.8 Informations relatives à l'identification du DTM, du Type d'Équipement du DTM et du matériel

4.8.1 DTM et Type d'Équipement du DTM

Un DTM prend en charge deux services afin de demander des informations qui lui sont propres et relatives aux types d'équipements pris en charge (voir la Figure 20). En général, ces informations sont utilisées par l'Application-Cadre pour créer des bibliothèques, choisir un DTM lors de la génération de la topologie des FDT ou pour de simples validations.

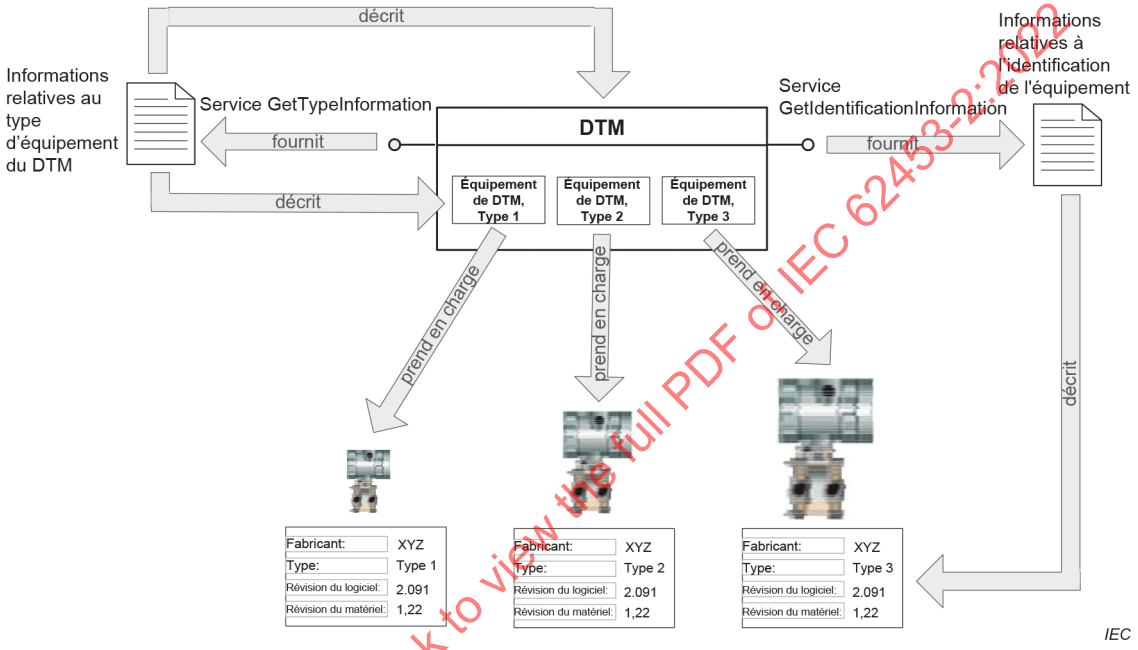


Figure 20 – Informations relatives à l'identification du DTM, du Type d'Équipement du DTM et de l'équipement

Un DTM désigne un composant logiciel comportant le logiciel d'application spécifique à un équipement. Le service GetTypeInformation (voir 7.2.3.1) retourne les informations générales relatives à cette partie du logiciel telles que le nom, la version et le fournisseur du logiciel, la version du FDT (voir également 4.13), ainsi qu'une liste des types d'équipements pris en charge.

La représentation d'un type d'équipement particulier au sein du DTM est dénommée Type d'équipement du DTM. Un DTM peut contenir un ou plusieurs Types d'Équipements du DTM. La conception et la mise en œuvre réelles des Types d'Équipements du DTM ne relèvent pas du domaine d'application de la spécification des FDT, mais le service GetTypeInformation retourne également des informations sur les parties du logiciel telles que le nom, la version, le fournisseur, les protocoles pris en charge, etc.

Lorsqu'une Application-Cadre ajoute un DTM à la topologie des FDT, elle sélectionne alors un des Types d'Équipements du DTM pris en charge (voir service Initialize (Initialiser), 7.2.4.1). Le DTM doit conserver la sélection au sein de ses données d'instance, pour pouvoir la restituer lorsqu'un DTM représentant le même équipement est initialisé à l'occurrence suivante. Le DTM doit fournir des informations relatives au Type d'Équipement du DTM sélectionné par le service GetActiveTypeInfo (voir 7.2.3.4). Le service doit toujours retourner le Type d'Équipement du DTM actuel. En cas de mise à jour du DTM, le nouveau DTM doit retourner les informations relatives au Type d'Équipement du DTM mises à jour.

4.8.2 Identification du matériel pris en charge

Les informations relatives aux types d'équipements physiques pouvant être gérées par les Types d'Équipements du DTM sont retournées par le service GetIdentificationInformation (voir 7.2.3.2). Par exemple, le fabricant, le type, la version du matériel et du logiciel intégré de l'équipement. Le DTM peut même retourner des caractères génériques pour certains éléments spécifiques d'identification de l'équipement pour signaler que le Type d'Équipement du DTM peut être utilisé pour tous les équipements avec lesquels le caractère générique concorde (par exemple, le caractère astérisque "*" pour la version du matériel peut signaler que le Type d'Équipement du DTM prend en charge toutes les versions du matériel).

Les informations retournées par le service GetIdentificationInformation sont spécifiques au bus de terrain et sont par conséquent définies dans le document IEC 62453-3xy qui décrit l'intégration des profils de protocole dans les FDT. Cependant, un FDT définit les méthodes de conversion des informations dans un format indépendant du protocole pour permettre aux Applications-Cadres sans connaissance spécifique au protocole de les utiliser. La conversion dépend de la technologie de mise en œuvre et par conséquent, est définie dans les documents relatifs au profil d'intégration des modèles d'ojets de la série IEC 62453 qui décrivent le mapping avec des technologies de mise en œuvre particulières.

4.8.3 Identification du matériel connecté

Un DTM prend par ailleurs en charge le service HardwareInformation (voir 7.2.3.3) qui permet de lire des informations relatives à l'équipement en ligne à partir de l'équipement connecté (voir la Figure 21).

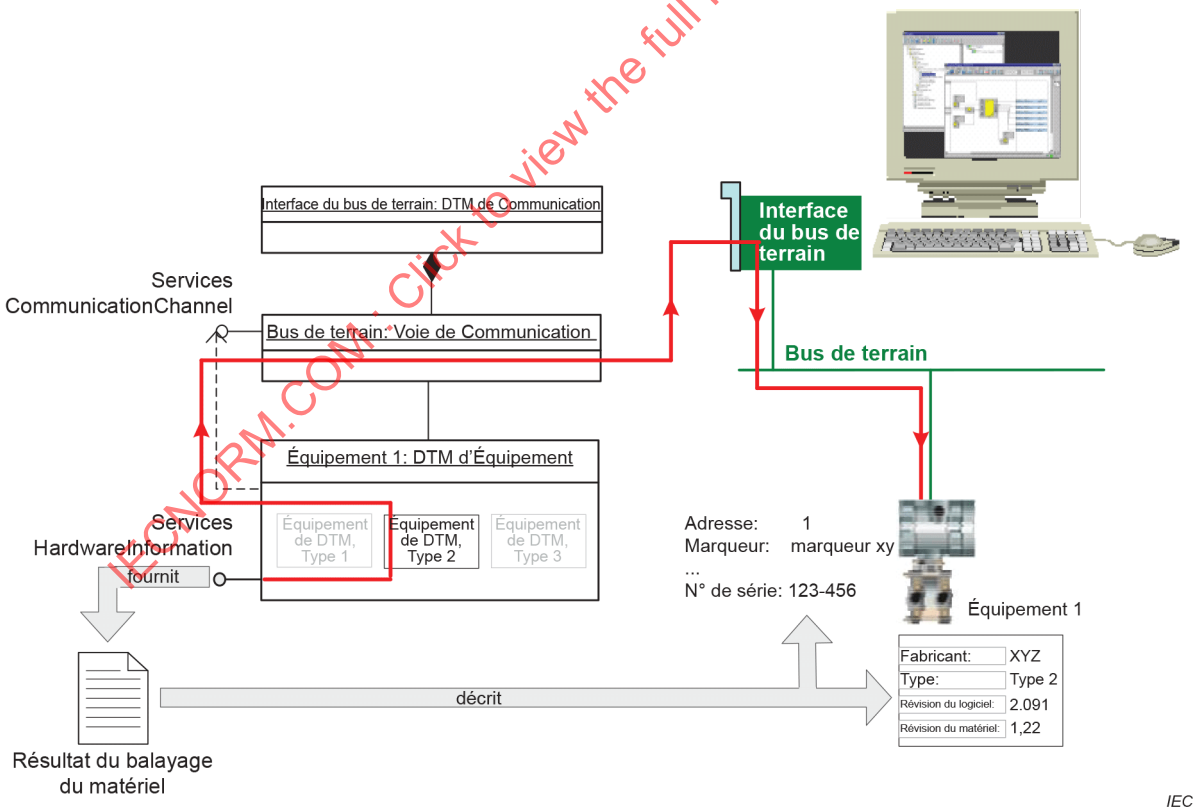


Figure 21 – Identification du matériel connecté

Le service retourne les informations relatives au type d'équipement telles que le fabricant, le type, la version du matériel et du logiciel intégré ainsi que les informations relatives aux instances de l'équipement telles que l'adresse, le marqueur et le numéro de série du bus de terrain. Ces informations sont également spécifiques au bus de terrain, comme les informations retournées par le service GetIdentificationInformation (voir 7.2.3.2). Par conséquent, le format réel et la conversion dans un format indépendant du protocole sont également définis dans le document IEC 62453-3xy qui décrit l'intégration des profils de protocole dans les FDT. Voir les exemples en 6.2.4 et en 6.2.5.

4.9 Persistance et synchronisation des données du DTM

Il existe trois types fondamentaux de données relatives au DTM:

- les données relatives aux instances (appelées "données d'instance" ("instance data")). Les données relatives aux instances appartiennent au DTM proprement dit. Il incombe au DTM de choisir les données qu'il stocke, mais il doit garantir qu'il est capable de représenter l'instance de l'équipement stockée en chargeant ces données;
- les données qui ne sont pas relatives aux instances. Ces données appartiennent à un Projet, sont globales ou spécifiques au Type d'Équipement du DTM;
- des blocs de données. Des données spécifiques au DTM, par exemple des données historiques, des données temporaires.

Le format de tous les types de données est spécifique au DTM. Cependant, un DTM doit présenter un Identificateur Unique Universel qui spécifie le format utilisé pour sauvegarder ses données d'instance. De plus, un DTM doit fournir une liste des formats de données compatibles qu'il est capable de charger. Ces identificateurs de format sont inclus dans les informations relatives au Type d'Équipement du DTM retournées par le service GetTypeInformation (voir 7.2.3.1). Une Application-Cadre peut utiliser les identificateurs de format pour retrouver le DTM correct après une mise à niveau du logiciel du DTM (voir 8.9).

Un FDT définit deux types de mécanismes de persistance des données du DTM. Un DTM doit utiliser soit le stockage du Projet de l'Application-Cadre, soit un stockage privé du DTM (voir la Figure 22).

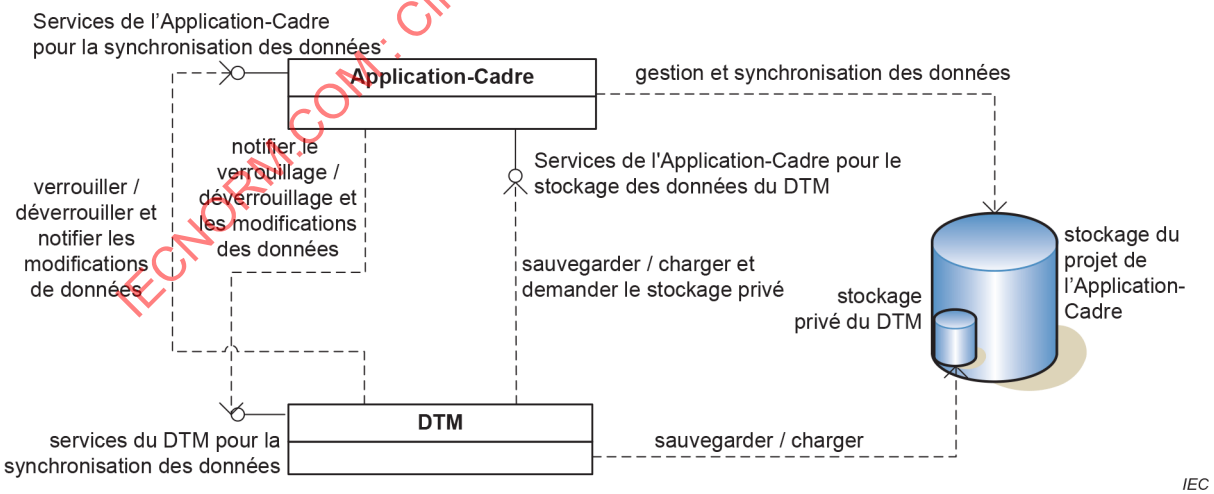


Figure 22 – Mécanismes de synchronisation et de stockage des FDT

Les services LoadInstanceData (voir 7.7.5.1) et SaveInstanceData (voir 7.7.5.2) permettent au DTM de sauvegarder et de charger les données relatives aux instances dans le stockage du Projet de l'Application-Cadre. L'Application-Cadre doit garantir la cohérence des données pour un accès aux données multiutilisateur et multiclient. Le DTM doit utiliser les 'services de l'Application-Cadre pour la synchronisation des données du DTM' (voir 7.7.6) pour tenter de verrouiller ses données relatives aux instances avant leur modification. L'Application-Cadre doit utiliser les 'services du DTM pour la synchronisation des données' (voir 7.2.13) pour avertir un DTM des événements, par exemple si les données ont été verrouillées par une autre instance du DTM. Voir 8.5.

Le service GetPrivateDtmStorageInformantion (voir 7.7.5.3) retourne des informations relatives au stockage privé du DTM auxquelles il peut directement accéder sans nécessiter l'utilisation d'autres services fournis par l'Application-Cadre. Le stockage privé du DTM permet à celui-ci de stocker des données relatives aux instances, des données qui ne sont pas relatives aux instances ainsi que des blocs de données. Le DTM doit lui-même garantir la cohérence des données pour un accès aux données multiutilisateur et multiclient. L'Application-Cadre est chargée d'élaborer une stratégie de sauvegarde pour les stockages privés du DTM.

Un DTM doit pouvoir fonctionner sans effets secondaires si le stockage privé du DTM n'est pas disponible. Il doit garantir qu'il n'y a aucune incidence sur les données d'instance gérées par les services SaveInstanceData (voir 7.7.5.1) et LoadInstanceData (voir 7.7.5.2).

Le mécanisme de stockage privé du DTM dépend de la technologie de mise en œuvre et est ainsi défini dans les documents relatifs au profil d'intégration des modèles d'ojets de la série IEC 62453 qui définissent le mapping avec des technologies particulières.

4.10 Accès aux paramètres de l'équipement du DTM

Un DTM prend en charge les services de lecture et d'écriture des paramètres de l'équipement stockés dans les données d'instance du DTM (voir 7.2.8) et directement dans l'équipement connecté (voir 7.2.10).

Les paramètres disponibles de l'équipement sont spécifiques à un DTM et dépendent du type d'équipement et de bus de terrain. Les informations sémantiques (voir SemanticId dans le Tableau A.4) sont transmises aux paramètres de l'équipement. Ainsi, leur utilisation peut par exemple permettre à une Application-Cadre d'obtenir les informations relatives à la signification et à l'utilisation d'un seul paramètre ou de la structure des données. La définition relative à l'identificateur est spécifique à un protocole, elle est par conséquent définie dans les spécifications IEC 62453-3xy qui définissent l'intégration des profils de protocole dans les FDT.

4.11 Diagramme d'états d'un DTM

4.11.1 États d'un DTM

Le diagramme suivant (voir la Figure 23) définit les différents états d'un DTM.

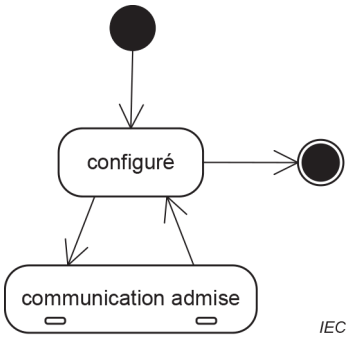


Figure 23 – Diagramme d'états d'un DTM

Le diagramme d'états se compose de deux états: 'configuré' et 'communication admise'. Les services du DTM disponibles dans un état donné sont définis dans les descriptions des services (voir l'Article 7).

L'Application-Cadre déclenche toutes les transitions d'états en appelant les services du DTM correspondants. La transition d'état est réussie si le service retourne fdt:serviceSucceeded = TRUE. Le DTM conserve son état précédent, si la transition d'état est rejetée en retournant fdt:serviceSucceeded = FALSE (FAUX). Les conditions dans lesquelles le DTM est autorisé à rejeter les transitions d'état sont définies dans les descriptions des services.

Transition d'État: État initial – état 'configuré'

Le service Initialize (voir 7.2.4.1) doit être appelé pour faire passer le DTM de son état initial à l'état 'configuré'.

Transition d'État: État 'configuré' – état 'communication admise'

Le service EnableCommunication (voir 7.2.4.3) doit être appelé avec TRUE pour faire passer le DTM de l'état 'configuré' à l'état 'communication admise'. Condition préalable pour cet appel: l'Application-Cadre informe le DTM de l'infrastructure de communication en appelant le service SetLinkedCommunicationChannel (voir 7.2.4.2).

Seul cet état permet au DTM d'établir une liaison de communication avec son équipement en appelant le service Connect (voir 7.6.1.2) de sa Voie de Communication reliée. Les sous-états de cet état sont décrits dans le diagramme d'états de communication (voir 4.11.2).

Transition d'État: État 'communication admise' – état 'configuré'

Le service EnableCommunication (voir 7.2.4.3) doit être appelé avec FALSE pour faire passer le DTM de l'état 'communication admise' à l'état 'configuré'.

Transition d'État: État 'configuré' – état final

Le service Terminate (voir 7.2.4.6) doit être appelé pour faire passer le DTM de l'état 'configuré' à son état final.

Le Tableau 3 présente une vue d'ensemble de ces transitions.

Tableau 3 – Transitions des états du DTM

État de départ	État final	Déclencheur	Condition
État initial	Configuré	Initialize (Initialiser)	
Configuré	Communication admise	EnableCommunication(TRUE)	Le DTM dipose de l'infrastructure de communication
Communication admise	Configuré	EnableCommunication(FALSE)	
Configuré	État final	Terminate (Mettre fin)	

4.11.2 Sous-états de 'Communication admise'

Ce diagramme d'états décrit les sous-états de l'état du DTM 'communication admise' (voir la Figure 24).

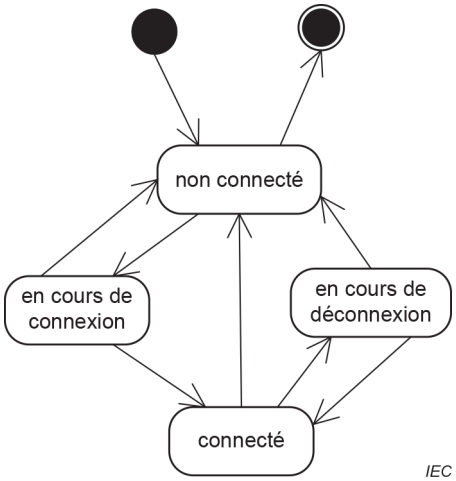


Figure 24 – Sous-états de communication admise

Le Tableau 4 décrit les transitions des sous-états de 'communication admise' du DTM.

Tableau 4 – Transitions des sous-états de 'communication admise' du DTM

État de départ	État final	Déclencheur	Condition	Action
'non connecté'	'en cours de connexion'	Déclencheur du service Online (En ligne) du DTM		Demande de connexion
'en cours de connexion'	'non connecté'		La connexion ne peut pas être établie	
'en cours de connexion'	'connecté'	Réponse de connexion		
'connecté'	'non connecté'	Arrêt prématuré de la connexion par la communication		
'connecté'	'en cours de déconnexion'	Le Service Online du DTM est terminé		Service Déconnexion
'en cours de déconnexion'	'connecté'		La connexion ne peut pas faire l'objet d'un arrêt prématuré	
'en cours de déconnexion'	'déconnecté'	Réponse de déconnexion		

4.12 Phases d'exploitation de base

4.12.1 Rôles et droits d'accès

Lorsqu'un DTM est lancé par l'Application-Cadre, le rôle de l'utilisateur est transmis au DTM qui doit restreindre l'accès aux paramètres conformément au rôle.

De même, la disponibilité des fonctions et l'aspect des interfaces utilisateur peuvent être adaptés.

Exemples:

- un observateur n'a pas accès aux fonctions d'étalonnage;
- un observateur ne doit pas modifier les paramètres.

Voir 5.2.

4.12.2 Phases d'exploitation

Si une Application-Cadre demande des fonctions disponibles au DTM, elle exécute la phase d'exploitation, qui doit être utilisée par un DTM pour adapter la disponibilité des fonctions et l'aspect de l'interface utilisateur.

Il existe cinq phases d'exploitation:

- Ingénierie
  - planification et configuration d'une installation industrielle,
  - pas d'accès en ligne à l'installation industrielle;
- Mise en service, atelier
  - installation de l'infrastructure de l'installation industrielle et des équipements,
  - balayage du réseau pour vérifier un réseau planifié,
  - téléchargement des données du projet dans l'installation industrielle,
  - programmation, diagnostic des équipements,
  - réglage des paramètres;
- Durée d'exécution
  - l'installation industrielle a été totalement mise en service et fonctionne,
  - accès très restreint aux données de configuration et de paramétrage,
  - balayage pour vérifier le réseau,
  - remplacement des équipements défectueux,
  - lecture des valeurs de processus et des informations relatives au diagnostic;
- Service
  - cette phase d'exploitation est utilisée pour les Applications-Cadres de l'outil des services.
    Balayage pour générer une topologie. Balayage pour vérifier un réseau. Toutes les opérations relatives aux services. Libre accès à l'équipement;
- Pas prise en charge
  - l'application ne prend pas en charge les phases d'exploitation définies ci-dessus,
  - un DTM doit proposer toutes les fonctions si l'Application-Cadre passe par la phase "Absence de prise en charge".

Le Tableau 5 décrit l'utilisation des phases d'exploitation dans différents types d'Applications-Cadres.

Tableau 5 – Phases d'exploitation

Application-Cadre du Système	Application-Cadre de l'outil des services	Autres Applications-Cadres
Ingénierie	Service	notSupported
Mise en service		
Durée d'exécution		

Par exemple, pendant la phase de mise en service, le DTM peut autoriser l'acteur de maintenance à effectuer le paramétrage de l'ensemble complet des paramètres (accès au paramétrage en ligne). Pendant la phase d'exécution, il peut toutefois l'empêcher ou l'autoriser seulement pour certains de ses paramètres.

4.13 Interopérabilité des versions du FDT

4.13.1 Vue d'ensemble de l'interopérabilité des versions

Un FDT est un concept fondé sur un composant, qui est en constante évolution vers des versions améliorées. En revanche, les environnements de système de commande fonctionnent généralement pendant une période comprise entre 10 ans et 15 ans. Une combinaison de composants conçus pour différentes versions du FDT peut apparaître si les composants du matériel et du logiciel doivent être échangés dans un système de commande.

Ces dispositions soulèvent la question du mode de coopération possible des composants fondés sur différentes versions du FDT.

Le paragraphe 4.13 traite essentiellement de deux sujets concernant l'interopérabilité des versions du FDT:

a) Persistance

De nouvelles versions des composants FDT (Applications-Cadres et DTM) doivent être capables de charger des données d'instance de versions plus anciennes.

b) Interopérabilité des composants

L'interopérabilité des composants de différentes versions du FDT (dans le cadre d'une seule technologie de mise en œuvre) doit être adaptée. Les DTM de versions du FDT plus récentes doivent fonctionner dans les Applications-Cadres plus anciennes et inversement. La communication doit fonctionner même si les versions du FDT des DTM d'Équipement et des DTM de Communication sont différentes. L'interaction entre ces composants dépend de la technologie et est par conséquent définie de manière détaillée dans les documents relatifs au profil d'intégration des modèles d'ojets.

Les futures améliorations des FDT sont subordonnées au facteur contraignant d'assurer une compatibilité maximale des différentes versions du FDT. Cette compatibilité entraîne l'optimisation de l'interopérabilité des composants FDT conçus à l'origine pour différentes versions du FDT.

L'interopérabilité des versions à deux niveaux différents est examinée dans les documents relatifs au profil d'intégration des modèles d'ojets de la série IEC 62453.

1) Niveau de spécification

La spécification du FDT vise une totale compatibilité des différentes versions de la norme. Les composants FDT correctement conçus sont totalement compatibles sans mises en œuvre supplémentaires.

2) Niveau de mise en œuvre

Dans les documents relatifs au profil d'intégration des modèles d'ojets, la version du FDT comprend un numéro principal, un numéro secondaire, un numéro de publication et un numéro d'élaboration (par exemple 1.2.0.3 où 1 est le numéro principal, 2 est le numéro secondaire, 0 celui de la publication et 3 le numéro d'élaboration).

La compatibilité est définie de manière légèrement différente selon le numéro principal:

– la compatibilité entre les composants FDT avec le même numéro principal de version du FDT est garantie par la spécification du profil d'intégration des modèles d'ojets;

– la compatibilité entre les composants FDT avec des numéros secondaires, de publication et d'élaboration de version du FDT différents peut être obtenue avec des chemins codés supplémentaires. La spécification des FDT fournit les informations et les mécanismes nécessaires pour prendre en charge une compatibilité totale à ce niveau. Le numéro secondaire ou de publication augmente si une nouvelle fonctionnalité, dépendante de son importance appropriée, est ajoutée à la

spécification des FDT. Le numéro d'élaboration augmente si une correction d'erreur dans la spécification ou la bibliothèque de types est nécessaire.

4.13.2 Versions du DTM et de l'équipement

Lorsqu'un nouveau DTM est fourni pour une nouvelle version supérieure des FDT, il est fortement recommandé d'utiliser de nouvelles informations d'enregistrement (voir 7.2.2). En cas de modification des informations d'enregistrement d'un DTM, les documents relatifs au profil d'intégration des modèles d'ojets de la série IEC 62453 fournissent un mécanisme pour les mettre à niveau.

Dans son catalogue d'équipements, une nouvelle version du DTM peut fournir la même liste ou un sous-ensemble des équipements de l'ancienne version du DTM. Dans ce cas, une Application-Cadre FDT gère les deux versions d'un DTM comme tout autre DTM capable de gérer le même dispositif de terrain comme deux DTM distincts.

Un DTM d'une version du FDT plus élevée qui n'utilise pas de nouvelles informations d'enregistrement doit prendre en charge au moins tous les Types d'Équipements du DTM pris en charge par les versions précédentes de ce DTM.

4.13.3 Persistance

En cas de modification de la version de l'Application-Cadre ou du DTM, la persistance (chargement des Projets stockés) peut alors représenter un problème.

Une Application-Cadre doit être capable de charger d'anciennes données d'instance et de fournir les données d'instance non modifiées au DTM même en cas de modification de la version de ce dernier. Un DTM doit être capable de charger d'anciennes données d'instance tant que ses informations d'enregistrement ne sont pas modifiées. Lorsque les informations d'enregistrement d'un DTM ont été modifiées, la spécification des FDT fournit un mécanisme dépendant -de la technologie pour les mettre à niveau au nouveau DTM défini dans les documents relatifs au profil d'intégration des modèles d'ojets. De cette manière, il est possible de charger d'anciennes données d'instance existantes d'un DTM dans un autre objet DTM. Le nouveau DTM est chargé de convertir les données d'instance en conséquence.

Un nouveau DTM doit fournir des informations relatives aux données d'instance de DTM qu'il peut charger (compatibilité de l'ensemble de données). Il doit prendre en charge tous les Types d'Équipements du DTM de l'ancien DTM. Si l'Application-Cadre reconnaît le nouveau DTM, elle peut en informer l'utilisateur (par exemple, "Un nouveau DTM compatible est installé, souhaitez-vous utiliser la nouvelle version à la place de l'ancienne?"). Si l'utilisateur confirme l'instanciation du nouvel objet DTM à la place de l'ancien, les anciennes données d'instance sont alors chargées et converties.

4.13.4 Communication imbriquée

4.13.4.1 Vue d'ensemble de la communication imbriquée

Étant donné que les DTM peuvent être attachés conformément à la topologie des FDT, il est nécessaire qu'ils communiquent correctement. Dans la mesure où les DTM concernés peuvent prendre en charge différentes versions du FDT, les restrictions suivantes de 4.13.4.2 à 4.13.4.4 doivent être prises en considération.

4.13.4.2 Échange d'informations

Les informations échangées par le biais des services de communication peuvent comporter des paramètres dépendants de la version.

Par conséquent, un DTM ne doit utiliser que les paramètres de service de communication conformément à la version du FDT de son composant parent. Ces informations relatives à la version du composant parent doivent être fournies dans les informations de réponse du service

Connect (voir 7.6.1.2). Un DTM ne doit ensuite utiliser que les paramètres de communication compatibles avec cette version du FDT.

4.13.4.3 Mise à niveau de la Voie de Communication

Une Voie de Communication doit prendre en charge les anciennes versions mineures du FDT si elle est mise à niveau vers une version mineure plus récente. Cette disposition s'explique par le fait que certains de ses enfants préalablement existants peuvent ne pas prendre en charge les interfaces de la version mineure plus récente.

4.13.4.4 Compatibilité des versions

S'agissant de la communication imbriquée, le numéro de version retourné par un DTM de Passerelle dépend de la fonctionnalité de son composant parent.

Si le numéro de la version du DTM de Passerelle est supérieur à celui de la version du composant parent, le DTM de Passerelle doit alors:

- retourner la même version que celle du parent et fournir la fonctionnalité conformément à cette version, ou
- reproduire la fonctionnalité supérieure si possible. Le DTM de Passerelle doit alors garantir que toutes les fonctionnalités exigées sont fournies par son parent.

5 Modèle de session et cas d'utilisation du FDT

5.1 Vue d'ensemble du modèle de session

L'Article 5 décrit les cas d'utilisation pris en considération lors de la conception de la spécification du FDT. Une Application-Cadre peut ne prendre en charge qu'un sous-ensemble de ces cas d'utilisation. De même, une Application-Cadre peut prendre en charge d'autres cas d'utilisation non définis dans cet Article 5.

La Figure 25 représente les principaux cas d'utilisation.

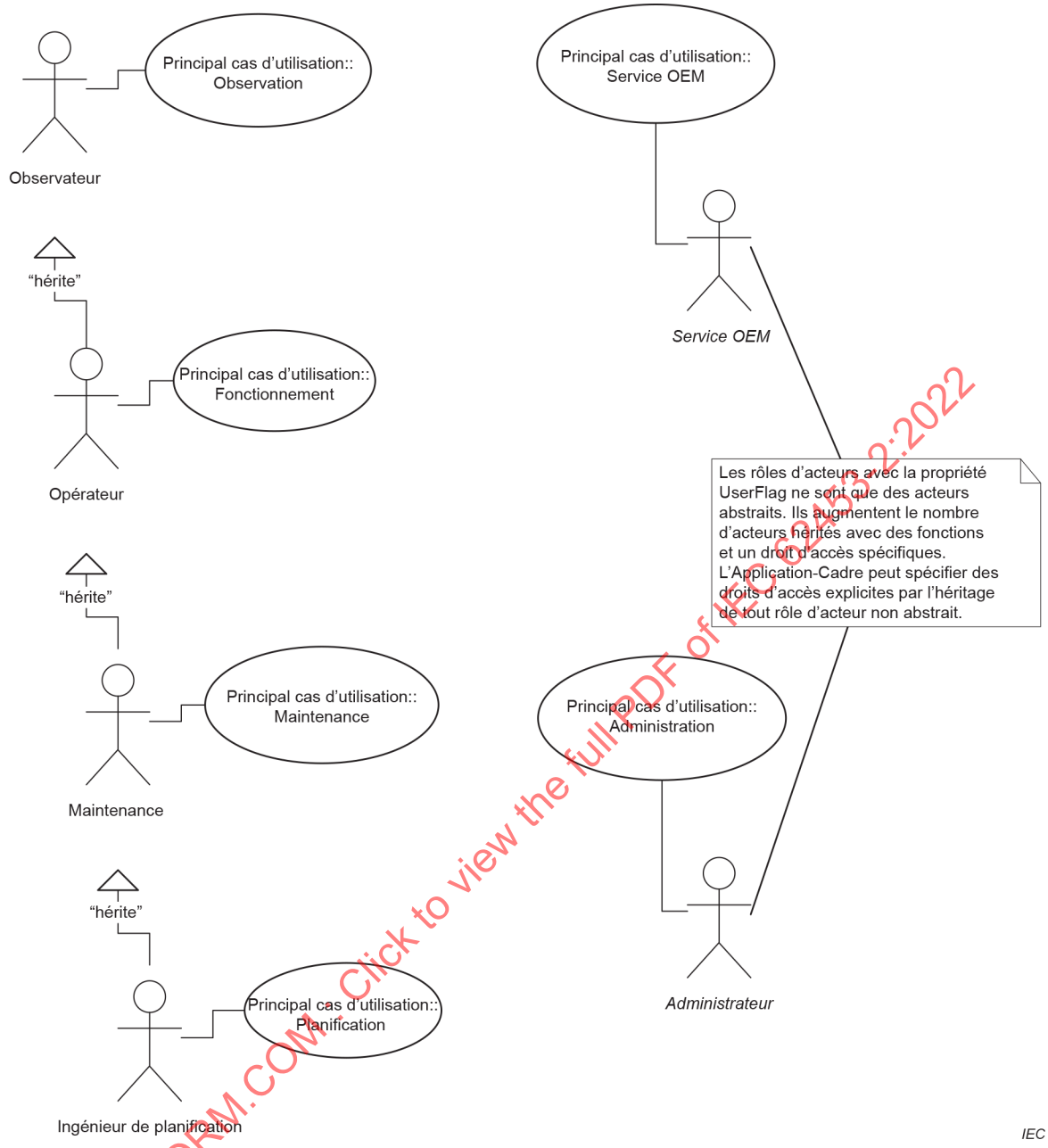


Figure 25 – Diagramme des principaux cas d'utilisation

Les principaux cas d'utilisation sont détaillés en 5.3, avec des descriptions textuelles et de manière facultative, certains scénarios nécessaires.

5.2 Acteurs

Les types d'acteurs représentés dans le diagramme des cas d'utilisation de la Figure 25 sont structurés de manière hiérarchique.

L'acteur "Maintenance" ("Maintenance") hérite de l'acteur "Opérateur" ("Operator"). L'acteur "Ingénieur de planification" ("Planning Engineer") hérite des cas d'utilisation de l'acteur "Maintenance". Les acteurs "Administrateur" ("Administrator") et "Service OEM" ("OEM Service") peuvent augmenter le nombre d'acteurs hérités avec d'autres cas d'utilisation. Les cas d'utilisation, qui sont transmis à un acteur de plus haut niveau, peuvent agir dans une plus large mesure pour correspondre à leur objet.

Le Tableau 6 décrit les rôles des acteurs de manière succincte:

Tableau 6 – Acteurs

Nom de l'acteur:	Observateur
Description succincte:	Cet acteur désigne une personne qui ne peut qu'observer le processus actuel
Hérite:	Aucun
Niveau de l'utilisateur:	FDTUserInformation.userLevel = observer
Vérification de l'accès:	L'accès en tant qu'acteur "Observateur" peut ne pas nécessiter de mot de passe
Cas d'utilisation:	Aucun
Nom de l'acteur:	**Opérateur**
Description succincte:	Cet acteur désigne une personne qui doit observer et gérer le processus actuel. L'"Opérateur" peut donc vérifier le statut actuel de l'équipement, modifier des valeurs fixées et vérifier si l'équipement fonctionne correctement. Il convient que les cas d'utilisation pour ce rôle d'acteur permettent à l'utilisateur de réaliser un diagnostic complet, d'observer le statut réel et l'ensemble de paramètres, ainsi que les variables de processus actuelles
Hérite:	Observateur
Niveau de l'utilisateur:	FDTUserInformation.userLevel = operator
Vérification de l'accès:	L'accès en tant qu'"Opérateur" nécessite un mot de passe
Cas d'utilisation:	Connexion de l'utilisateur (User r Login), Vue en ligne (Online View), Piste de vérification (Audit Trail), Archivage (Archive), Génération de rapport (Report Generation), Gestion des biens (Asset Management)
Nom de l'acteur:	**Maintenance**
Description succincte:	Il convient qu'une personne de "Maintenance"puisse réaliser toutes les opérations de maintenance nécessaires y compris l'échange d'équipement, la formation, l'étalonnage, le réglage, etc. La personne peut donc télécharger les ensembles de paramètres vérifiés, modifier un sous-ensemble de paramètres en ligne ou hors ligne, réaliser des opérations en ligne spécifiques à l'équipement et, en fin de traitement, peut avoir la possibilité de charger l'ensemble de paramètres complet
Hérite:	Opérateur
Niveau de l'utilisateur:	FDTUserInformation.userLevel = maintenance
Vérification de l'accès:	L'accès en tant qu'acteur de "Maintenance" exige un mot de passe
Cas d'utilisation:	Simulation, Fonctionnement en ligne (Online Operation), Réparation (Repair), Fonctionnement hors ligne (Offline Operation), Traitement des blocs de données (Bulk Data Handling) Connexion de l'utilisateur [a], Vue en ligne [a], Piste de vérification [a], Archivage [a], Génération de rapport [a], Gestion des biens[a]
Nom de l'acteur:	**Ingénieur de planification**
Description succincte:	L'acteur "ingénieur de planification" ("Planning Engineer") désigne une personne telle qu'un ingénieur des installations industrielles, un spécialiste (par exemple, conformément au document VDI/VDE2187) ou toute personne dûment autorisée. Cette personne a accès à l'ensemble complet des fonctions de l'Application-Cadre et peut utiliser les fonctions du DTM sans aucune restriction. Seules les opérations spécifiques à l'OEM sont verrouillées
Hérite:	Maintenance
Niveau de l'utilisateur:	FDTUserInformation.userLevel = planningEngineer
Vérification de l'accès:	L'accès en tant qu'acteur "ingénieur de planification" ("Planning Engineer") exige un mot de passe
Cas d'utilisation:	Traitement des Instances du DTM Simulation [a], Fonctionnement en ligne [a], Réparation[a], Fonctionnement hors ligne [a], Traitement des blocs de données [a], Connexion de l'utilisateur [a], Vue en ligne [a], Piste de vérification [a], Archivage [a], Génération de rapport [a], Gestion des biens[a]

Nom de l'acteur:	Administrateur (acteur abstrait)
Description succincte:	L'acteur "Administrateur" ("Administrator") désigne une personne qui doit réaliser des opérations administratives au sein d'un environnement d'ingénierie Il est responsable de l'ajout et de la suppression de composants du logiciel
Hérite:	Dépend de l'Application-Cadre
Fanion de l'utilisateur [b]:	FDTUserInformation.administrator = TRUE
Vérification de l'accès:	L'accès en tant qu'acteur "Administrateur" exige un mot de passe
Cas d'utilisation:	Intégration et tous les cas d'utilisation hérités
Nom de l'acteur:	**Service OEM (acteur abstrait)**
Description succincte:	L'acteur "Service OEM" ("OEM Service") peut réaliser l'ensemble complet des fonctions spécifiques au DTM. L'opération spécifique à l'OEM peut être accessible à un acteur "Service OEM", comme la réinitialisation des compteurs internes et les échanges de microprogrammes. L'acteur "Service OEM" ne comporte que des cas d'utilisation hérités, mais ces cas (notamment, le cas d'utilisation "Fonctionnement en ligne") peuvent avoir de véritables extensions.
Hérite:	Dépend de l'Application-Cadre
Fanion de l'utilisateur [b]:	FDTUserInformation.oemService = TRUE
Vérification de l'accès:	La vérification de l'accès à un "Service OEM" doit avoir deux niveaux de sécurité: 1. Mot de passe de l'Application-Cadre: permet l'accès à la fonctionnalité de l'Application-Cadre 2. Mot de passe spécifique à l'équipement: permet l'accès à l'opération OEM avec l'équipement. La vérification du mot de passe spécifique à l'équipement revient au DTM ou même à l'équipement proprement dit
Cas d'utilisation:	Cas d'utilisation hérités seulement

[a] Cas d'utilisation hérités.

[b] "Fanions de l'utilisateur" ("User Flags") peut être combiné avec n'importe quel "Niveau de l'utilisateur" ('User level'), pour spécifier le rôle de l'acteur hérité d'un acteur "Administrateur" ou "Service OEM".

5.3 Cas d'utilisation

5.3.1 Vue d'ensemble des cas d'utilisation

Le paragraphe 5.3 décrit tous les cas d'utilisation du diagramme des cas d'utilisation sous la forme d'un tableau. Tous les attributs des cas d'utilisation inclus, qui sont identiques au cas d'utilisation principal associé, ne sont pas affichés, afin de réduire le volume d'informations.

5.3.2 Observation

Figure 26 – Cas d'utilisation "Observation"

Seuls les cas d'utilisation "Observation" peuvent être exécutés par l'acteur "Observateur" (voir la Figure 26). L'acteur "Observateur" désigne une personne ayant le niveau d'accès le plus bas. Aucun cas d'utilisation n'est associé à cette personne.

5.3.3 Réalisation des activités opérationnelles

Les cas d'utilisation "Fonctionnement" ('Operation') sont disponibles pour l'acteur "Opérateur" (voir la Figure 27).

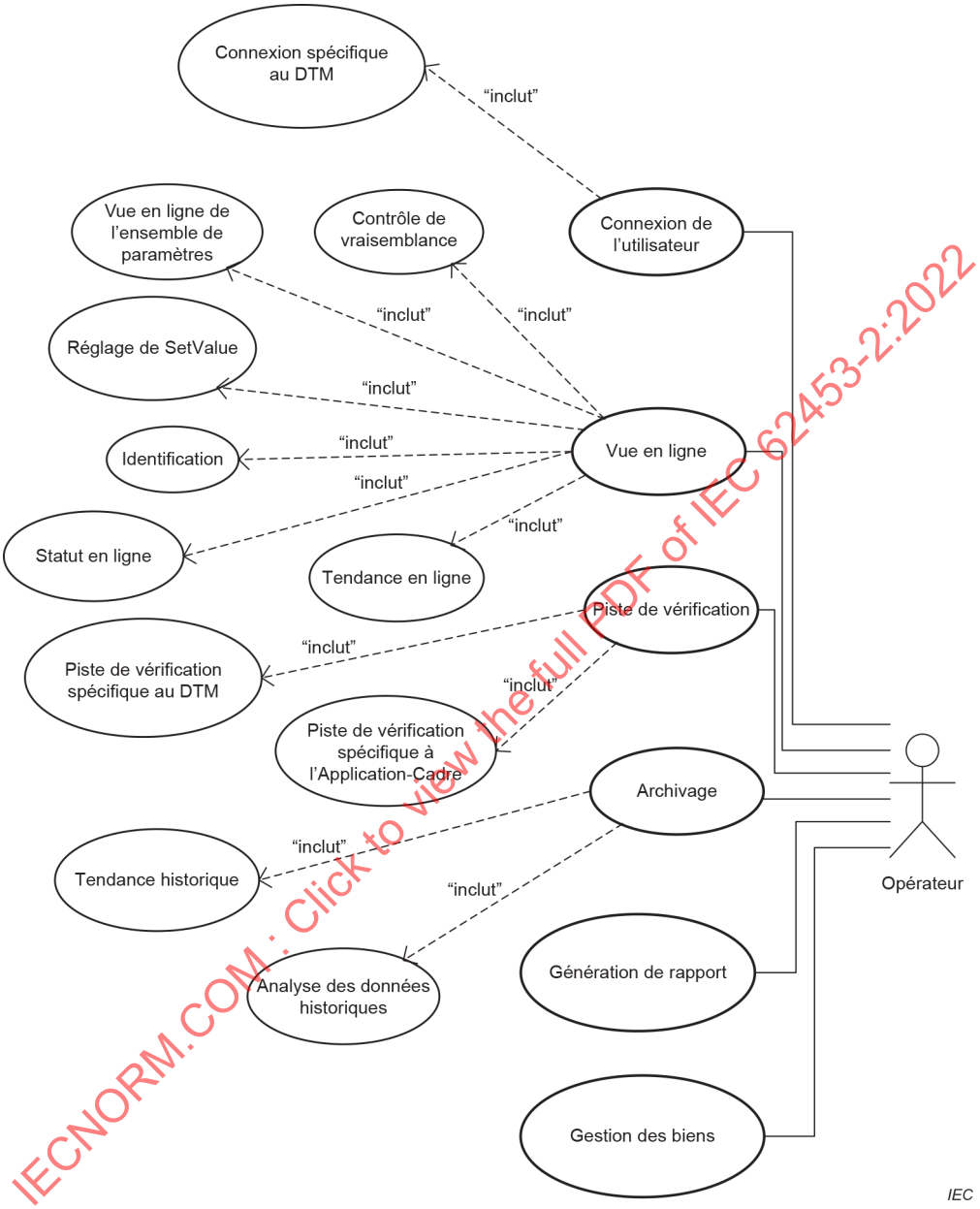


Figure 27 – Cas d'utilisation "Fonctionnement"

Le Tableau 7 décrit les cas d'utilisation "Fonctionnement".

Tableau 7 – Cas d'utilisation "Fonctionnement"

Cas d'utilisation:		**Connexion de l'utilisateur**
Description succincte:		Une personne d'un type d'acteur spécifique se connecte à l'Application-Cadre. En cas d'échec de la vérification, la personne ne peut agir que comme un acteur "Observateur"
IUG:		Partie de l'Application-Cadre
Impression:		Pas de prise en charge
Connexion à l'équipement:		Non établie
Niveau de l'utilisateur	Observateur:	Dépend de l'Application-Cadre
	Opérateur:	Accessible
	Maintenance:	
	Ingénieur de planification:	
UserFlag	Administrateur:	
	Service OEM:	Accessible avec une fonctionnalité étendue (voir ci-dessous)
Cas d'utilisation inclus:		**Connexion spécifique au DTM**
Description succincte:		Accès aux informations spécifiques au fabricant, par exemple les codes de production, le journal des modifications
IUG:		Partie du DTM
Impression:		Pas de prise en charge
Connexion à l'équipement:		Nécessite généralement une connexion établie
UserFlag	Service OEM:	Accessible uniquement pour le Service OEM
Cas d'utilisation:		**Vue en ligne**
Description succincte:		Ce cas d'utilisation offre un ensemble complet d'opérations pour visualiser les paramètres et le statut d'équipement
IUG:		L'interface utilisateur graphique (IUG) est partie intégrante du DTM. L'Application-Cadre peut proposer des vues en ligne pour un groupe d'équipements
Impression:		Il convient que la vueen ligne prenne toujours en charge la fonction d'impression pour produire la documentation
Connexion à l'équipement:		Nécessite une connexion établie à un ou plusieurs équipements (Le réglage de SetValue peut influencer les données de l'équipement)
Niveau de l'utilisateur	Observateur:	Pas d'accès
	Opérateur:	Accessible
	Maintenance:	
	Ingénieur de planification:	
UserFlags:		Aucune modification
Cas d'utilisation inclus:		**Vue en ligne de l'ensemble des paramètres**
Description succincte:		Permet à l'acteur de charger les paramètres à partir de l'équipement pour la visualisation et la documentation (impression). (applicationId du DTM: fdtObserve)
Cas d'utilisation inclus:		**Réglage de SetValue**
Description succincte:		Permet à l'acteur de régler les SetValues d'un équipement (par exemple, le positionneur, les contrôleurs). (applicationId du DTM: fdtAdjustSetValue)
Cas d'utilisation inclus:		**Statut en ligne**
Description succincte:		Cas d'utilisation pour visualiser et analyser le statut actuel de l'équipement (applicationId du DTM: fdtDiagnosis)
Cas d'utilisation inclus:		**Tendance en ligne**
Description succincte:		Cas d'utilisation pour générer et surveiller les tendances des valeurs dynamiques en ligne
Cas d'utilisation inclus:		**Contrôle de vraisemblance**
Description succincte:		À chaque modification des paramètres de l'équipement ou des données de configuration, un contrôle de vraisemblance est effectué automatiquement. Tant que le contrôle de vraisemblance échoue, les données ne peuvent pas être saisies dans l'équipement

Cas d'utilisation inclus:		Identification
Description succincte:		Affiche les informations relatives à l'identification de l'instance de l'équipement, par exemple, l'adresse, le marqueur (TAG), la révision du bus de terrain (FieldBus-Rev), la révision du DD (DD-Rev), le microprogramme (Firmware). Ces informations dépendent du protocole. Elles peuvent également comporter les informations spécifiques à un type d'équipement. D'autres informations peuvent être fournies: références à la documentation, zone pour ajouter des remarques à l'instance de l'équipement. Se reporter à l'article Identification de l'équipement dans les spécifications de FDT spécifiques à un protocole. (applicationId du DTM: fdtIdentify)
Cas d'utilisation:		**Piste de vérification**
Description succincte:		Cas d'utilisation pour permettre à l'acteur de visualiser et de supprimer les données de la piste de vérification
IUG:		Le composant, qui prend en charge la piste de vérification, doit fournir une interface utilisateur graphique adéquate
Impression:		Impression à des fins de documentation nécessaire pour les pistes de vérification et doit donc être prise en charge
Connexion à l'équipement:		Aucune connexion nécessaire
Niveau de l'utilisateur	Observateur:	Pas d'accès
	Opérateur:	Accessible pour visualisation seulement
	Maintenance:	
	Ingénieur de planification:	Visualiser, exporter et supprimer
UserFlags:		Aucune modification
Cas d'utilisation inclus:		**Piste de vérification spécifique au DTM**
Description succincte:		Chaque DTM peut lui-même prendre en charge une piste de vérification. (applicationId du DTM: fdtAuditTrail)
Cas d'utilisation inclus:		**Piste de vérification spécifique à l'Application-Cadre**
Description succincte:		L'Application-Cadre peut mettre en œuvre une piste de vérification globale du système en utilisant des données internes et des propriétés nommées du DTM exportées du DTM par le biais des services de la piste de vérification
Cas d'utilisation:		**Archivage**
Description succincte:		Le cas d'utilisation "Archivage" décrit l'accès aux archives de l'Application-Cadre pour la visualisation et l'analyse des données
IUG:		Le composant, qui prend en charge des fonctions d'archivage, doit fournir une interface utilisateur graphique adéquate
Impression:		Il convient que chaque composant d'archive prenne également en charge l'impression
Connexion à l'équipement:		Pas nécessaire
Niveau de l'utilisateur	Observateur:	Pas d'accès
	Opérateur:	Accessible pour visualisation seulement
	Maintenance:	
	Ingénieur de planification:	Visualiser, exporter et supprimer
UserFlags:		Aucune modification
Cas d'utilisation inclus:		**Tendance historique**
Description succincte:		Les opérations d'archivage peuvent prendre en charge la visualisation de données historiques (par exemple, la tendance)
Cas d'utilisation inclus:		**Analyse des données historiques**
Description succincte:		Certaines Applications-Cadres et certains DTM peuvent prendre en charge des opérations étendues pour analyser les données historiques
Cas d'utilisation:		**Génération de rapport**
Description succincte:		La fonction d'impression d'un cas d'utilisation peut normalement être lancée à partir de son interface utilisateur graphique (IUG). Outre l'impression de la vue réelle d'une IUG, certaines Applications-Cadres peuvent mettre en œuvre un mécanisme de documentation configurable qui permet à l'acteur de générer des rapports. Ce rapport peut être généré en combinant les impressions de plusieurs cas d'utilisation ou de plusieurs équipements. Par exemple, un rapport peut être généré et contenir les impressions de "Vue en ligne", "Piste de vérification" et "Fonctionnement en ligne" pour un équipement. Un rapport peut également être généré et contenir la configuration d'un élément d'infrastructure de communication et de ses clients
IUG:		Application-Cadre
Impression:		Application-Cadre en combinaison avec un ou plusieurs DTM
Connexion à l'équipement:		Dépend du type de rapport
UserLevel, UserFlag:		Les informations imprimées peuvent dépendre du niveau de l'utilisateur et des fanions de l'utilisateur

Cas d'utilisation:		Gestion des biens
Description succincte:		L'Application-Cadre fournit un mécanisme pour la gestion des biens
IUG:		Le composant, qui prend en charge les fonctions de gestion des biens, doit fournir une interface utilisateur graphique adéquate. (applicationId du DTM: fdtAssetManagement)
Impression:		Il convient que l'Application-Cadre qui prend en charge ce cas d'utilisation prenne également en charge l'impression.
Connexion à l'équipement:		Pas nécessaire
Niveau de l'utilisateur	Observateur:	Pas d'accès
	Opérateur:	Accessible pour visualisation seulement
	Maintenance:	
	Ingénieur de planification:	Visualiser, exporter et supprimer
UserFlags:		Aucune modification

5.3.4 Maintenance

Les cas d'utilisation "Maintenance" sont disponibles pour l'acteur "Ingénieur de maintenance" (“Maintenance Engineer”) (voir la Figure 28).

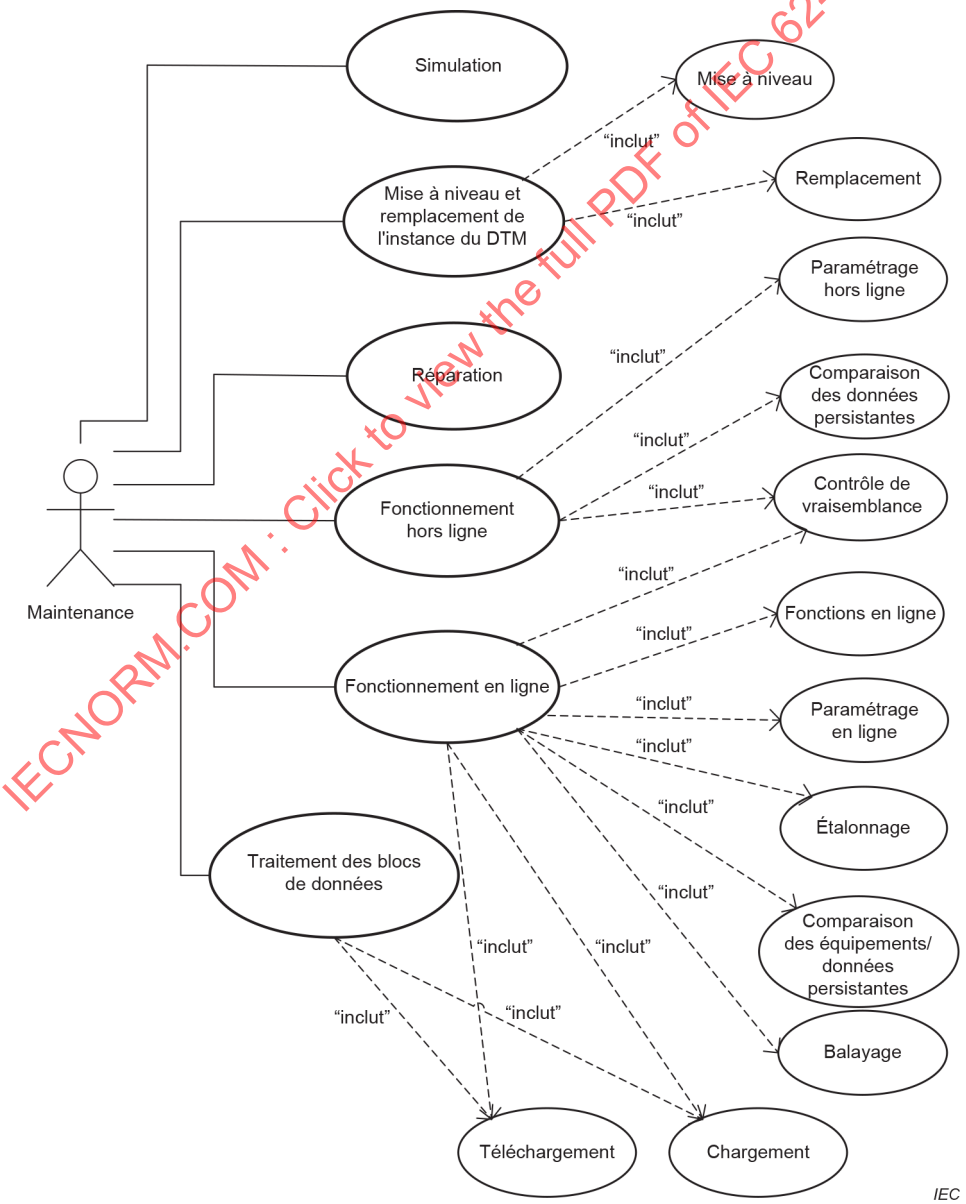


Figure 28 – Cas d'utilisation "Maintenance"

Les cas d'utilisation "Maintenance" sont décrits dans le Tableau 8.

Tableau 8 – Cas d'utilisation "Maintenance"

Cas d'utilisation:		**Simulation**
Description succincte:		Ce cas d'utilisation simule le comportement d'un équipement. Par conséquent, l'acteur est autorisé à apporter des modifications temporaires au sein des paramètres de l'équipement, comme forcer l'équipement à établir une sortie analogique de courant fixe
IUG:		Spécifique à un DTM
Impression:		Spécifique à un DTM
Connexion à l'équipement:		Doit avoir une connexion établie
Niveau de l'utilisateur	Observateur:	Pas d'accès
	Opérateur:	
	Maintenance:	Accessible
	Ingénieur de planification:	
UserFlags:		Aucune modification
Cas d'utilisation:		**Opération hors ligne**
Description succincte:		Ce cas d'utilisation inclut toutes les opérations qui n'exigent pas de connexion établie avec un équipement. Uniquement pour le téléchargement montant ou descendant, une connexion à un équipement peut être temporairement établie
IUG:		DTM et Application-Cadre
Impression:		DTM
Connexion à l'équipement:		Dépend du cas d'utilisation inclus
Niveau de l'utilisateur	Observateur:	Pas d'accès
	Opérateur:	
	Maintenance:	Accessible
	Ingénieur de planification:	
UserFlags:		Aucune modification
Cas d'utilisation inclus:		**Contrôle de vraisemblance**
Description succincte:		À chaque modification des valeurs des paramètres, un contrôle de vraisemblance est automatiquement effectué. Tant que le contrôle de vraisemblance échoue, les données ne peuvent pas être sauvegardées dans la base de données ou saisies dans l'équipement. Il peut y avoir deux options selon les règles ou l'environnement d'application: Après affichage d'un avertissement, les données incohérentes peuvent être stockées Pour des raisons de sécurité, après affichage d'un avertissement, la saisie des données est interdite
Utilisateurs:		Le contrôle de vraisemblance peut être plus tolérant pour les acteurs de plus haut niveau
Cas d'utilisation inclus:		**Paramétrage hors ligne**
Description succincte:		L'utilisateur peut modifier les valeurs des paramètres. Les modifications n'affectent que les données dans la base de données de l'Application-Cadre après avoir été stockées comme des données persistantes. Il convient que les données soient signalées comme étant des données transitoires jusqu'à ce qu'elles soient stockées
IUG:		DTM (applicationId: fastOfflineParametrize)
Impression:		DTM
Connexion à l'équipement:		Aucune connexion nécessaire
Cas d'utilisation inclus:		**Comparaison des données persistantes**
Description succincte:		Permet à l'utilisateur de comparer l'ensemble des paramètres hors ligne avec les ensembles de paramètres d'autres instances, d'ensembles de paramètres par défaut de l'équipement ou du Projet. Les ensembles de données peuvent être modifiables et les données peuvent être transférées d'un ensemble de données à l'autre
IUG:		DTM (applicationId: fastOfflineCompare)
Impression:		Peut être prise en charge par le DTM
Connexion à l'équipement:		Pas nécessaire

Cas d'utilisation:		Réparation
Description succincte:		Ce cas d'utilisation représente les opérations qui doivent être effectuées pour réparer ou modifier un équipement. Exemple: Une Application-Cadre prend en charge le retrait temporaire d'un équipement avec paramétrage du téléchargement automatique lorsque l'équipement est réinstallé
Niveau de l'utilisateur	Observateur:	Pas d'accès
	Opérateur:	
	Maintenance:	Accessible
	Ingénieur de planification:	
UserFlag	Administrateur:	Aucune modification
	Service OEM:	Accessible avec une fonctionnalité étendue (voir ci-dessous)
Cas d'utilisation:		**Mise à niveau et remplacement de l'instance du DTM**
Description succincte:		Ce cas d'utilisation représente les opérations qui doivent être effectuées pour mettre à niveau ou remplacer un DTM. La mise à niveau ou le remplacement est utilisé(e) lorsque le nouveau DTM diffère du précédent. Le format des données du nouveau DTM peut être soit compatible (mise à niveau) soit incompatible (remplacement) avec le format des données du DTM précédent
Niveau de l'utilisateur	Observateur:	Pas d'accès
	Opérateur:	
	Maintenance:	Accessible
	Ingénieur de planification:	
UserFlag	Administrateur:	Aucune modification
	Service OEM:	Aucune modification
Cas d'utilisation inclus:		**Remplacement**
Description succincte:		Ce cas d'utilisation représente des opérations qui doivent être effectuées pour remplacer un DTM dans le cas d'un format d'un ensemble de données incompatible
Cas d'utilisation inclus:		**Mise à niveau**
Description succincte:		Ce cas d'utilisation représente des opérations qui doivent être effectuées pour mettre à niveau un DTM dans le cas d'un format d'un ensemble de données compatible
Cas d'utilisation:		**Fonctionnement en ligne**
Description succincte:		Ce cas d'utilisation inclut toutes les opérations qui sont effectuées directement avec l'équipement
IUG:		Spécifique à un DTM
Impression:		Spécifique à un DTM
Connexion à l'équipement:		Connecté ou déconnecté
Utilisateurs:	Opérateur:	Pas d'accès
Niveau de l'utilisateur	Observateur:	Pas d'accès
	Opérateur:	
	Maintenance:	Accessible
	Ingénieur de planification:	Totalement accessible, à l'exclusion du paramètre spécifique au service OEM
UserFlag	Administrateur:	Aucune modification
	Service OEM:	Accessible avec une fonctionnalité étendue (voir ci-dessous)
Cas d'utilisation inclus:		**Fonctions en ligne**
Description succincte:		Le cas d'utilisation "Fonctions en ligne" offre toutes les procédures qui incluent la communication directe avec l'équipement. Le cas d'utilisation peut inclure des procédures simples comme la réinitialisation, mais aussi des procédures plus complexes avec plusieurs sections comme l'étalonnage ou l'apprentissage
Cas d'utilisation inclus:		**Paramétrage en ligne**
Description succincte:		Lorsque ce cas d'utilisation est lancé, tous les paramètres de l'équipement sont lus et présentés à l'utilisateur au sein d'une IUG. Au sein de l'interface utilisateur graphique, l'utilisateur peut modifier les paramètres, selon le niveau de l'utilisateur. L'équipement est mis à jour immédiatement si le paramétrage modifié est dans un état vérifié. (applicationId du DTM: fdtOnlineParameterize)
Cas d'utilisation inclus:		**Comparaison des données persistantes/données de l'équipement**
Description succincte:		Ce cas d'utilisation permet à l'utilisateur de comparer les données de l'équipement et les données persistantes. L'utilisateur peut modifier les données et les copier entre ces deux sources. (applicationId du DTM: fdtOnlineCompare)

Cas d'utilisation inclus:		**Étalonnage**
Description succincte:		Ce cas d'utilisation invoque les procédures d'étalonnage pour régler l'entrée pour, par exemple, les transmetteurs de pression ou le courant pour la sortie. (applicationId du DTM: fdtCalibration)
Cas d'utilisation inclus:		**Contrôle de vraisemblance**
Description succincte:		À chaque modification des paramètres de l'équipement ou des données de configuration, un contrôle de vraisemblance est effectué automatiquement. Tant que le contrôle de vraisemblance échoue, les données ne peuvent pas être saisies dans l'équipement.
UserLevel, UserFlag:		Le contrôle de vraisemblance peut être plus tolérant pour les acteurs de plus haut niveau
Cas d'utilisation inclus:		**Téléchargement montant**
Description succincte:		Lit l'ensemble des paramètres de l'équipement dans la base de données de l'Application-Cadre. Un chargement lancé par un acteur "Service OEM" peut charger d'autres paramètres OEM
IUG:		Si l'Application-Cadre prend en charge le chargement et le téléchargement pour un groupe d'équipements, l'Application-Cadre peut proposer une IUG pour améliorer la commande du processus de téléchargement montant
Impression:		Pas de prise en charge
Connexion à l'équipement:		Nécessite une connexion temporaire
Cas d'utilisation inclus:		**Téléchargement descendant**
Description succincte:		Comme le téléchargement montant, mais dans le sens contraire
IUG:		Si l'Application-Cadre prend en charge les téléchargements montant et descendant pour un groupe d'équipements, l'Application-Cadre peut proposer une IUG pour améliorer la commande du processus de téléchargement descendant
Impression:		Pas de prise en charge
Connexion à l'équipement:		Nécessite une connexion temporaire
Cas d'utilisation inclus:		**Balayage**
Description succincte:		Extrait la liste des équipements disponibles à partir d'un objet Voie (Channel). L'objet Voie peut être fourni par un DTM ou par l'Application-Cadre
IUG:		L'Application-Cadre peut proposer une IUG pour la représentation de la liste d'équipements
Impression:		Pas de prise en charge
Connexion à l'équipement:		Nécessite une connexion temporaire. (La voie doit avoir un accès en ligne)
Cas d'utilisation:		**Traitement des blocs de données**
Description succincte:		Ce cas d'utilisation représente la possibilité de gérer (par exemple, les téléchargements montant et descendant, le réglage des paramètres ou la génération de rapports) un groupe d'instruments en une seule fois. Ce cas d'utilisation traite les blocs de données au niveau du système
IUG:		Application-Cadre
Impression:		Pas prise en charge
Connexion à l'équipement:		Nécessite une connexion
Niveau de l'utilisateur	Observateur:	Pas d'accès
	Opérateur:	
	Maintenance:	Accessible
	Ingénieur de planification:	
UserFlags:		Aucune modification
Cas d'utilisation inclus:		**Chargement**
Description succincte:		Lit le paramétrage entier des équipements du groupe dans la base de données de l'Application-Cadre
Cas d'utilisation inclus:		**Téléchargement descendant**
Description succincte:		Comme le téléchargement montant, mais dans le sens contraire

5.3.5 Planification

Les cas d'utilisation "Planification" sont disponibles pour l'acteur "Ingénieur de planification" ("Planning Engineer") (voir la Figure 29).

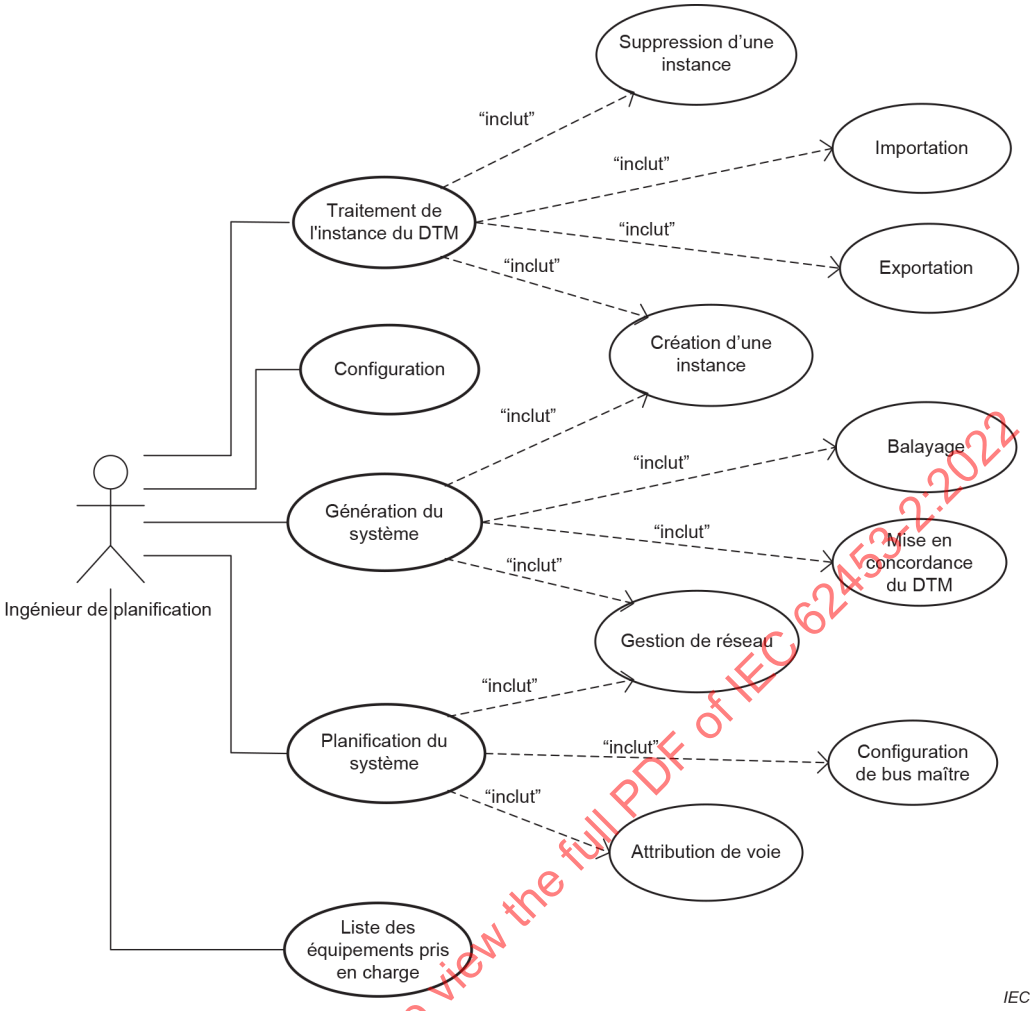


Figure 29 – Cas d'utilisation "Planification"

Le Tableau 9 décrit les cas d'utilisation "Planification".

Tableau 9 – Cas d'utilisation "Planification"

Cas d'utilisation:		**Traitement de l'instance du DTM**
Description succincte:		Ce cas d'utilisation permet à un acteur "Ingénieur de planification" de gérer (par exemple, instancier, importer, exporter, supprimer) une instance d'équipement dans l'Application-Cadre
IUG:		Application-Cadre et DTM
Impression:		Pas prise en charge
Connexion à l'équipement:		Pas nécessaire
Niveau de l'utilisateur	Observateur:	Pas d'accès
	Opérateur:	
	Maintenance:	
	Ingénieur de planification:	Accessible
UserFlag		Aucune modification
Cas d'utilisation inclus:		**Création d'une instance**
Description succincte:		Ce cas d'utilisation permet de créer une nouvelle instance d'équipement et de l'intégrer au Projet. Après la création d'une nouvelle instance d'équipement, l'ensemble des paramètres de l'équipement doit être instancié. Cette action est effectuée par le DTM
IUG:		Application-Cadre et DTM (s'il prend en charge la sélection par défaut des équipements)

Cas d'utilisation inclus:		**Importation**
Description succincte:		L'ensemble de paramètres peut être instancié par l'importation de données à partir d'une base de données (par exemple, une base de données modèle) ou par la copie de l'ensemble de paramètres à partir d'un équipement préalablement instancié et vérifié
IUG:		Application-Cadre
Cas d'utilisation inclus:		**Exportation**
Description succincte:		Pour copier les données d'une instance d'équipement à une autre, les données d'instance d'équipement peuvent être exportées
IUG:		Application-Cadre
Cas d'utilisation inclus:		**Suppression d'une instance**
Description succincte:		Suppression d'une instance d'équipement
IUG:		Application-Cadre
Cas d'utilisation:		**Configuration**
Description succincte:		Ce cas d'utilisation permet à l'acteur "Ingénieur de planification" de configurer un équipement complexe
IUG:		DTM (applicationId du DTM: fdtConfiguration)
Impression:		Peut être prise en charge
Connexion à l'équipement:		Ne doit pas avoir de connexion établie
Niveau de l'utilisateur	Observateur:	Pas d'accès
	Opérateur:	
	Maintenance:	
	Ingénieur de planification:	Accessible
UserFlags:		Aucune modification
Cas d'utilisation:		**Génération du système**
Description succincte:		Cas d'utilisation pour effectuer toutes les actions nécessaires à la génération de la topologie fondée sur le résultat du balayage
IUG:		Application-Cadre et DTM
Impression:		Application-Cadre et DTM
Connexion à l'équipement:		Nécessaire
Niveau de l'utilisateur	Observateur:	Pas d'accès
	Opérateur:	
	Maintenance:	
	Ingénieur de planification:	Accessible
UserFlags:		Aucune modification
Cas d'utilisation inclus:		**Balayage**
Description succincte:		Extrait la liste des équipements disponibles avec le balayage de service à partir d'un objet Voie. L'objet Voie peut être fourni par un DTM ou par l'Application-Cadre
IUG:		L'Application-Cadre peut proposer une IUG pour la représentation de la liste d'équipements
Impression:		Pas de prise en charge
Connexion à l'équipement:		Nécessite une connexion temporaire. (La voie doit avoir un accès en ligne)
Cas d'utilisation inclus:		**Gestion de réseau**
Description succincte:		Cas d'utilisation à des fins de gestion de réseau, par exemple, réglage d'adresse en fonction d'un DTM de Communication. (applicationId du DTM: fdtNetworkManagement)
Cas d'utilisation inclus:		**Mise en concordance du DTM**
Description succincte:		Cas d'utilisation pour identifier un DTM qui correspond le mieux aux informations extraites du cas d'utilisation "Balayage" ('Scan')
Cas d'utilisation inclus:		**Création d'une instance**
Description succincte:		Ce cas d'utilisation permet de créer une nouvelle instance d'équipement et de l'intégrer au Projet. Après la création d'une nouvelle instance d'équipement, l'ensemble des paramètres de l'équipement doit être instancié. Cette action est effectuée par le DTM
IUG:		Application-Cadre et DTM (s'il prend en charge la sélection par défaut des équipements)

Cas d'utilisation:		Planification du système
Description succincte:		Cas d'utilisation pour effectuer toutes les actions nécessaires à l'édition des informations relatives à la topologie.
IUG:		Application-Cadre et DTM
Impression:		Application-Cadre et DTM
Connexion à l'équipement:		Pas nécessaire
Niveau de l'utilisateur	Observateur:	Pas d'accès
	Opérateur:	
	Maintenance:	
	Ingénieur de planification:	Accessible
UserFlags:		Aucune modification
Cas d'utilisation inclus:		**Gestion de réseau**
Description succincte:		Cas d'utilisation à des fins de gestion de réseau, par exemple: réglage d'adresse en fonction d'un DTM de Communication. (applicationId du DTM: fdtNetworkManagement)
Cas d'utilisation inclus:		**Configuration de bus maître**
Description succincte:		Cas d'utilisation pour la configuration des DTM de bus maître
Cas d'utilisation inclus:		**Attribution de voie**
Description succincte:		Cas d'utilisation pour l'édition des attributions de voies du FDT
Cas d'utilisation:		**Liste des équipements pris en charge**
Description succincte:		Ce cas d'utilisation permet de visualiser et d'éditer une liste de tous les équipements pris en charge au sein d'un Projet. Un acteur "ingénieur de planification" peut sélectionner un équipement de cette liste pour créer une nouvelle instance d'équipement. Un acteur avec l'ensemble des fanions de l'acteur "Administrateur" dispose d'un accès pour modifier cette liste
IUG:		Application-Cadre
Impression:		Pas de prise en charge
Connexion à l'équipement:		Aucune connexion
Niveau de l'utilisateur	Observateur:	Pas d'accès
	Opérateur:	
	Maintenance:	
	Ingénieur de planification:	Accessible pour visualisation seulement
UserFlag	Administrateur:	Accessible pour édition/modification
	Service OEM:	Aucune modification

5.3.6 Service OEM

Figure 30 – Service OEM

Les cas d'utilisation "Service OEM" ('OEM Service') peuvent fournir à l'acteur "Service OEM" (voir la Figure 30) un accès étendu aux cas d'utilisation des rôles des autres acteurs.

5.3.7 Administration

Les cas d'utilisation "Administration" sont disponibles pour les utilisateurs qui ont des autorisations "Administrateur" supplémentaires (voir la Figure 31).

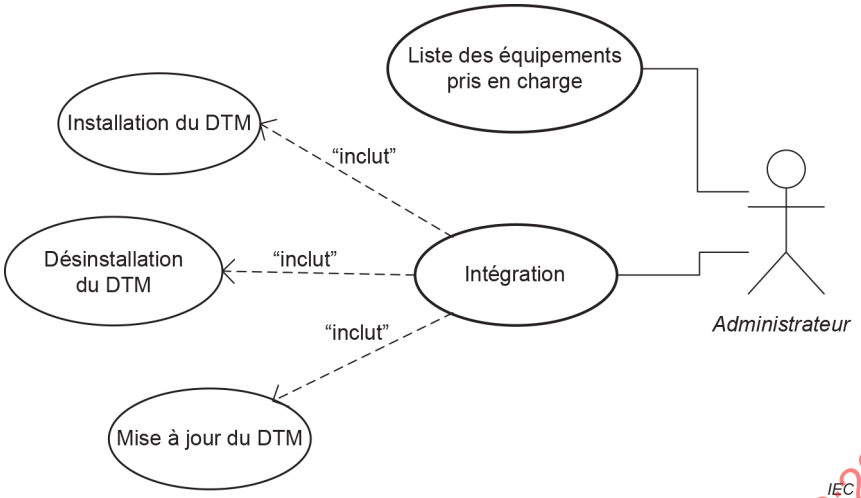


Figure 31 – Cas d'utilisation "Administrateur"

Le Tableau 10 décrit les cas d'utilisation "Administrateur".

Tableau 10 – Cas d'utilisation "Administrateur"

Cas d'utilisation:		**Intégration**
Description succincte:		Ce cas d'utilisation permet à l'acteur d'installer, de désinstaller ou de mettre à jour de nouveaux DTM. L'installation et la désinstallation ne sont pas contrôlées par l'Application-Cadre
IUG:		L'Application-Cadre peut prendre en charge la gestion des DTM
Impression:		Pas prise en charge
Connexion à l'équipement:		Ne doit pas avoir de connexion établie
Utilisateurs:	Opérateur:	Pas accessible
	Maintenance:	
	Ingénieur de planification:	
	Administrateur:	Accessible
	Service OEM:	Pas accessible
Cas d'utilisation inclus:		**Installation**
Description succincte:		Installation d'un nouveau DTM
IUG:		Responsabilité du DTM
Cas d'utilisation inclus:		**Désinstallation**
Description succincte:		Désinstallation d'un DTM
IUG:		Responsabilité du DTM
Cas d'utilisation inclus:		**Mise à jour du DTM**
Description succincte:		Mise à jour/modification d'une installation de DTM
IUG:		Responsabilité du DTM

6 Concepts généraux

6.1 Gestion des adresses

Les systèmes de bus de terrain utilisent généralement un concept d'adressage pour différencier les différents équipements connectés. Par conséquent, un DTM a généralement besoin des informations relatives à l'adresse de son équipement associé pour établir une connexion directe avec celui-ci (voir service Connect, 7.6.1.2).

Un DTM pour un type d'équipement qui peut être connecté à un système qui définit un concept d'adressage doit fournir les services NetworkManagementInfo (voir 7.2.11). Ces services permettent de lire et de saisir les informations relatives à l'adressage de l'équipement au DTM. Le schéma d'adressage est spécifique à un bus de terrain et par conséquent, les informations concrètes qui doivent être transmises au service sont définies par les documents IEC 62453-3xy qui décrivent l'intégration des profils de protocole dans les FDT.

Le composant qui fournit la Voie de Communication (DTM ou Application-Cadre) est toujours chargé du réglage de l'adresse dans les DTM reliés. Un DTM fournissant des Voies de Communication doit prendre en charge un objet Présentation qui permet à l'utilisateur de régler les adresses (ApplicationID = fdtNetworkManagement, voir le Tableau A.4) comme cela est représenté à la Figure 32.

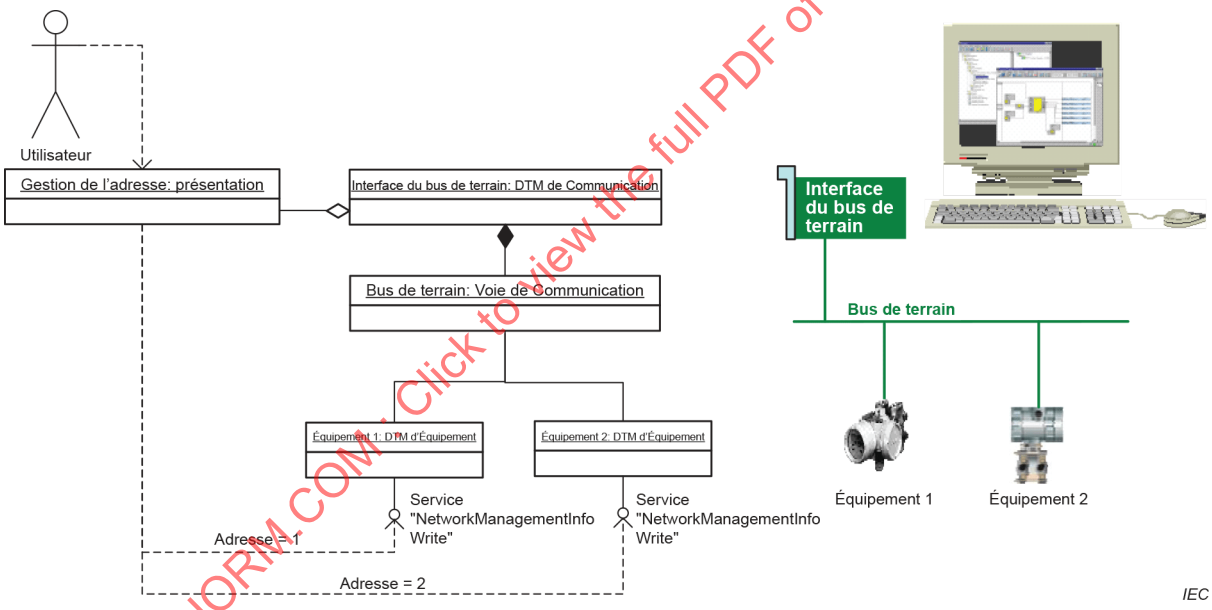


Figure 32 – Réglage d'adresse par le biais de l'objet Présentation du DTM

Par ailleurs, la Voie de Communication doit elle-même prendre en charge le service SetChildrenAdresses (voir 7.6.2.5) pour permettre à l'Application-Cadre de lancer le réglage d'adresse. Les diagrammes de séquences en 8.2 détaillent différents scénarios d'adressage.

La manière dont la gestion des adresses est assurée si la Voie de Communication est fournie par l'Application-Cadre ne relève pas du domaine d'application du présent document.

6.2 Balayage et attribution du DTM

6.2.1 Vue d'ensemble du balayage

De nombreux protocoles de bus de terrain (ou communication point à point) définissent la prise en charge des services de balayage pour demander une liste actualisée comportant les informations relatives aux équipements connectés. Les informations et services décrits en 6.2

permettent à l'Application-Cadre de générer ou de valider la topologie correspondante du FDT sans connaître les définitions spécifiques à un protocole.

6.2.2 Balayage

Le balayage est pris en charge par le service Balayage de la Voie de Communication (voir 7.6.4).

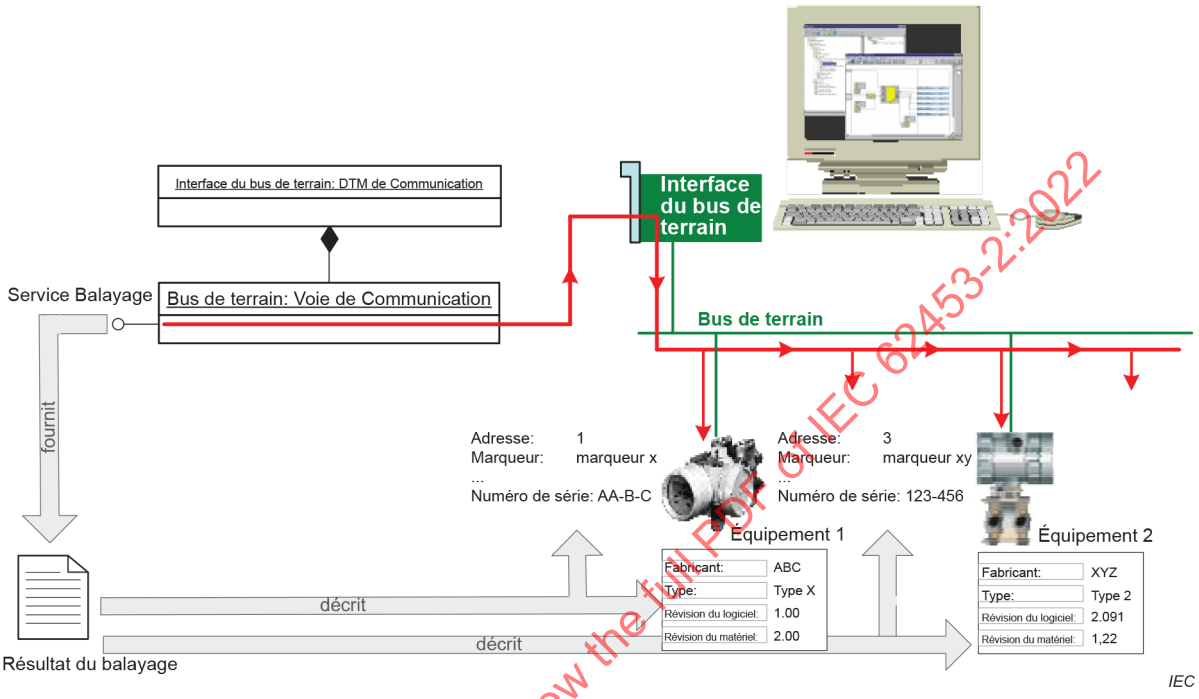


Figure 33 – Balayage du bus de terrain

Le service retourne les informations relatives aux instances et au type d'équipement pour tous les équipements connectés qui peuvent être déterminés par les moyens définis par le protocole de bus de terrain (voir la Figure 33). Par exemple, le fabricant de l'équipement, le type, la version du matériel et du logiciel intégré ou l'adresse, le marqueur et le numéro de série du bus de terrain. Ces informations sont spécifiques au bus de terrain. Le format concret et la conversion dans un format indépendant du protocole qui peuvent être utilisés par l'Application-Cadre sans connaissance spécifique du bus de terrain sont définis par le document IEC 62453- 3xy décrivant l'intégration des profils de protocole dans les FDT.

6.2.3 Attribution du DTM

Une Application-Cadre peut utiliser les informations retournées par un balayage (voir 7.6.4) afin d'identifier les DTM appropriés qui peuvent être ajoutés à la topologie des FDT en les comparant avec les informations d'identification du type d'équipement retournées par le service GetIdentificationInformation (voir 7.2.3.2) de DTM connus.

De plus, une Application-Cadre peut également utiliser les informations relatives aux résultats du balayage pour déterminer si les DTM dans une topologie existante des FDT sont appropriés ou non. Cette validation s'effectue en comparant les informations de résultats du balayage avec les informations d'identification du type d'équipement retournées par ces DTM. Chaque élément de ces informations issues du balayage peut être comparé avec les informations d'identification de l'équipement du DTM. Dans le cadre de cette comparaison, une Application-Cadre peut également appliquer des règles spécifiques.

6.2.4 Identification de l'équipement spécifique au fabricant

Parfois, les méthodes définies par le protocole du bus de terrain ne permettent pas d'identifier les équipements de manière unique. Par exemple, lorsque le même ID de type est utilisé pour plusieurs types d'équipements différents d'un fabricant. Dans ce cas, l'Application-Cadre peut identifier plusieurs Types d'Équipements du DTM ou même des DTM qui semblent appropriés pour les équipements identifiés lors d'un balayage. Cependant, un tel équipement peut être identifié par les méthodes définies par le fabricant de l'équipement.

Les FDT permettent au fabricant de mettre en œuvre cette identification spécifique de l'équipement au sein d'un DTM et permettent à une Application-Cadre de l'utiliser de façon normalisée, indépendamment du protocole et du fabricant.

Pour ce type d'équipement, le DTM doit ajouter les propriétés d'identification spécifiques au fabricant aux informations d'identification de l'équipement retournées par le service GetIdentificationInformation (voir 7.2.3.2). Si l'Application-Cadre tente d'identifier un DTM approprié, elle ne peut alors déterminer que les identifications d'équipements qui concordent, mais qui comportent des éléments supplémentaires. Les éléments supplémentaires ne sont pas inclus dans le résultat du balayage, car ils ne sont connus que du fabricant de l'équipement. Dans une telle situation, l'Application-Cadre doit rechercher un Type d'Équipement du DTM pour lequel DTMSupportLevel est réglé sur 'identSupport' (voir le Tableau A.10 idDTMSupportLevel du type de données). L''IdentSupport'-DTM est ajouté temporairement à la topologie des FDT afin de demander des informations complémentaires à l'équipement par le service HardwareInformation (voir 7.2.3.3). Le DTM doit alors exécuter l'identification de l'équipement spécifique -au fabricant et retourner les éléments supplémentaires d'identification de l'équipement résultant du service HardwareInformation. Cette opération permet à l'Application-Cadre d'effectuer une comparaison qui couvre toutes les informations d'identification de l'équipement, de présenter une concordance totale et de remplacer l''IdentSupport'-DTM par le DTM approprié et le Type d'Équipement de DTM concordant.

6.2.5 Balayage du matériel de communication

Les FDT définissent également les méthodes de balayage pour le matériel de communication qui agissent comme l'élément racine dans la topologie des FDT afin d'accéder au bus de terrain du niveau le plus élevé dans la topologie du système ou dans une liaison point à point.

Ce mécanisme repose sur un principe d'essais et d'erreurs. L'Application-Cadre instancie simplement chacun des DTM de Communication enregistrés ou présélectionnés, invoque le service HardwareInformation (voir 7.2.3.3) et vérifie si le service retourne des informations relatives au matériel de communication identifié ou une erreur.

6.3 Configuration du Maître du bus de terrain ou du Programmateur de Communication

De nombreux bus de terrain utilisent des équipements spécialisés pour contrôler la communication entre les différents participants, par exemple, le concept Maître-Esclave ou le concept de Passage du Jeton.

Les équipements qui contrôlent la communication sur le bus de terrain sont appelés Maître du bus de terrain ou Programmateur de Communication. Il est généralement nécessaire de configurer ces équipements avec les informations relatives aux paramètres du bus des équipements connectés au bus de terrain. En général, la configuration est réalisée à l'aide d'un outil de configuration spécifique à l'équipement. Le composant FDT qui représente le Maître du bus de terrain ou le Programmateur de Communication doit fournir cet outil de configuration. Cette disposition signifie que l'outil de configuration fait partie de l'Application-Cadre, ou, comme à la Figure 34, d'un DTM.

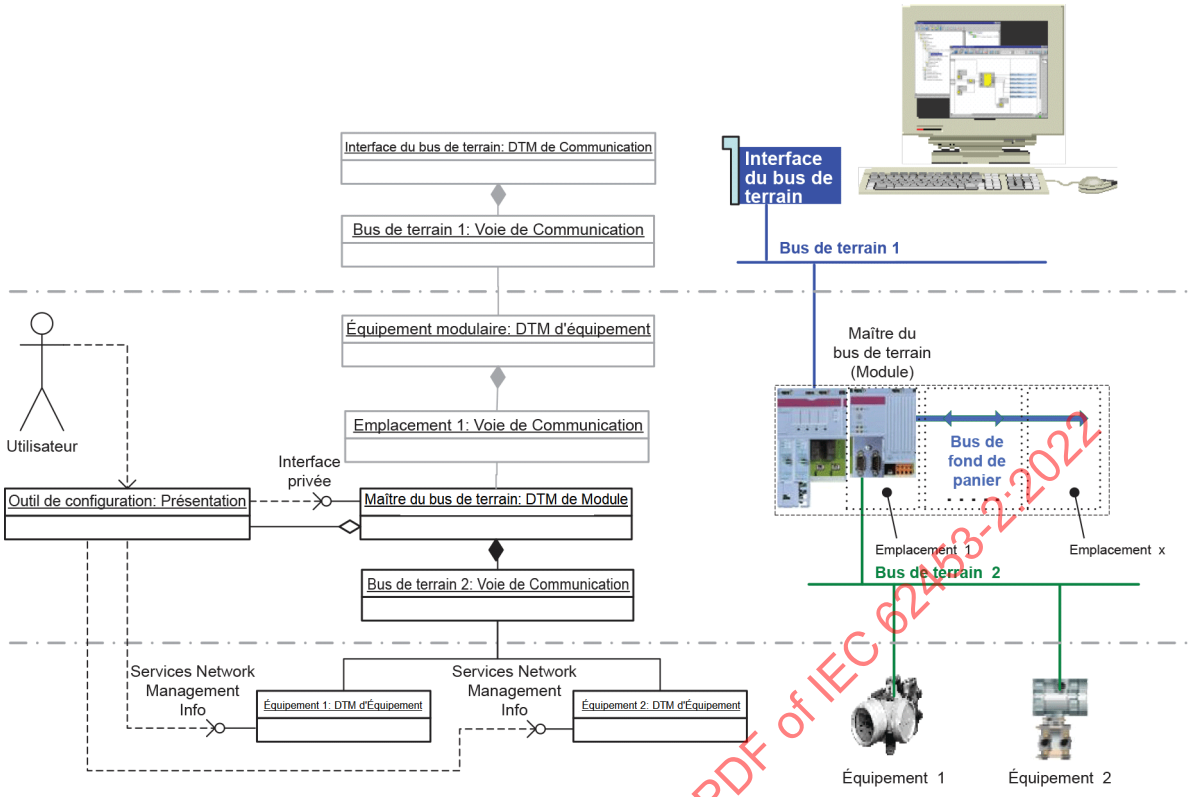


Figure 34 – Outil de configuration du maître du bus de terrain faisant partie d'un DTM

L'outil de configuration est capable de lire et de saisir les informations relatives aux paramètres du bus des participants du bus de terrain par le biais des services NetworkManagementInfo (voir 7.2.11) des DTM correspondants. Des méthodes privées définies entre l'outil de configuration et le DTM correspondant permettent d'accéder aux informations relatives aux paramètres de bus du maître ou du programmateur de communication.

Lorsque l'outil de configuration a défini toutes les informations relatives aux paramètres du bus, chaque DTM est chargé de saisir les informations dans son équipement par le biais des opérations de communication normalisées du FDT, par exemple si le service WriteDataToDevice demande un téléchargement descendant (voir 7.2.12.3).

Les informations spécifiques aux paramètres du bus pouvant être lues ou saisies par le biais des services NetworkManagementInfo sont définies dans les documents IEC 62453-3xy qui décrivent l'intégration des profils de protocole dans les FDT. De plus, la configuration générale du bus est également détaillée dans le document IEC 62453-3xy qui décrit le protocole.

6.4 Prise en charge de l'outil PLC

6.4.1 Généralités

Dans un système PLC type, un matériel de maître de bus de terrain transmet périodiquement les signaux E/S aux équipements connectés et les fournit au PLC dans une zone mémoire partagée. Cette zone mémoire est appelée image de processus (voir la Figure 35, simplifiée pour ne représenter que les données d'entrée). Les signaux E/S individuels sont traités par l'application PLC avec un décalage dans cette image, alors que la disposition globale de l'image est commandée par le logiciel de configuration du maître de bus de terrain qui peut être un DTM.

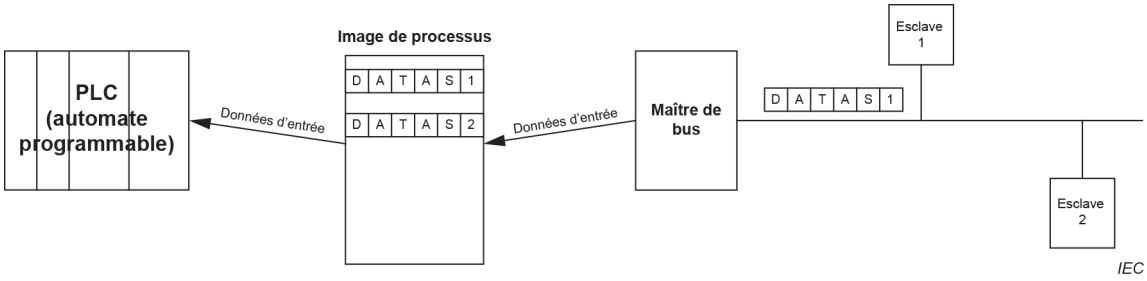


Figure 35 – Image de processus

Un DTM qui représente le maître de bus de terrain doit fournir les services permettant de configurer l'image de processus. Ces services permettent à une Application-Cadre d'outil PLC de demander des informations relatives à l'image de processus au DTM Maître de bus de terrain (voir la Figure 36). Ces informations servent au mapping des variables au sein d'un programme PLC avec les signaux E/S dans l'image de processus.

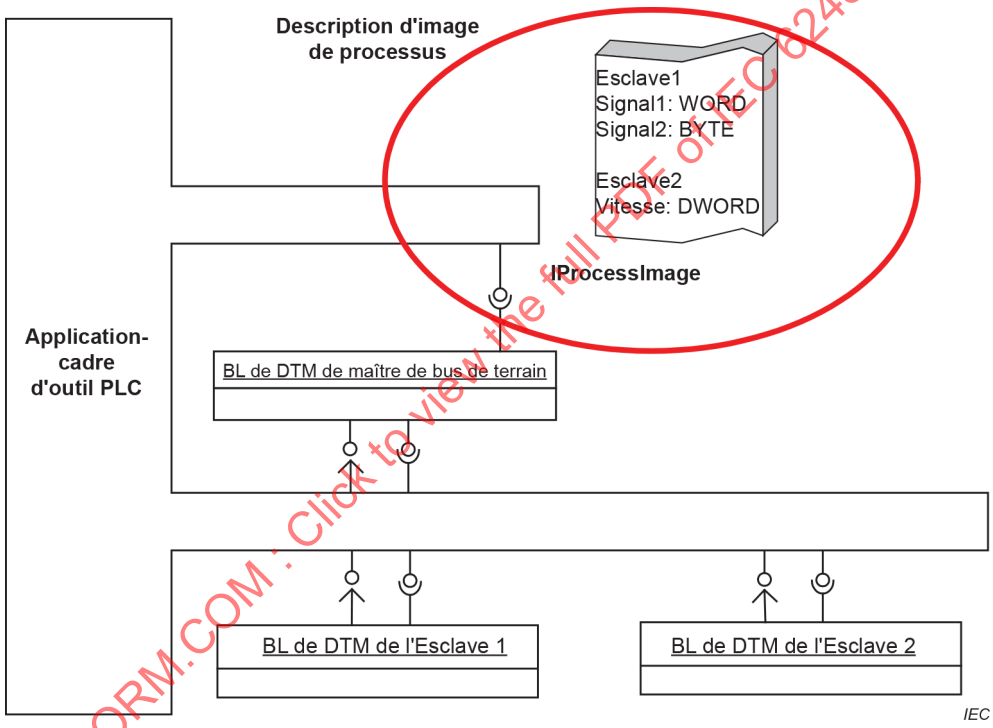


Figure 36 – Transfert d'informations relatives à la disposition à l'aide des services ProcessImage

La représentation de base des informations relatives aux images de processus est neutre par rapport au bus de terrain. Cette neutralité permet aux Applications-Cadres d'outil PLC d'attribuer les variables indépendamment du système de bus de terrain sous-jacent. En outre, les informations relatives aux signaux E/S spécifiques au bus de terrain (voir 6.4.2) sont également incluses dans les informations relatives aux images de processus. Ces informations sont rassemblées par le DTM Maître de bus de terrain à partir des DTM Enfants qui représentent les équipements connectés à l'aide des services ProcessData des DTM Enfants (voir 4.2.4.2).

6.4.2 Modifications des images de processus pendant le fonctionnement du PLC

Certains systèmes d'automatisation exigent de modifier la disposition de l'image de processus pendant que le PLC fonctionne (exécute une commande), car le processus de l'installation ne peut pas être arrêté.

Il est nécessaire que les Applications-Cadres d'outil PLC qui prennent en charge de telles modifications pendant la durée d'exécution déterminent si une modification peut être apportée sans arrêter le PLC. Autrement dit, si un utilisateur modifie une configuration qui peut impliquer de modifier les informations du réseau des DTM Enfants (voir 7.2.11), il convient alors de vérifier cette modification avant de l'appliquer au dispositif de terrain respectif. Cette disposition permet à l'Application-Cadre d'outil PLC de refuser les modifications qui exigent l'arrêt du PLC si cet arrêt n'est pas possible au même moment.

La nécessité de l'arrêt éventuel du PLC ne peut être validée que dans le DTM Maître de bus de terrain et dans l'Application-Cadre d'outil PLC elle-même. Ainsi, il convient que le DTM Enfant prenne en charge la validation des modifications apportées aux informations réseau en appelant les services NetworkInfoValidation de son parent. Le DTM Maître de bus de terrain peut alors redéfinir l'image de processus résultante et déterminer si elle peut être appliquée à l'aide des services ProcessImageValidation mis en œuvre par une Application-Cadre d'outil PLC (voir 7.7.7).

Il est nécessaire d'effectuer la validation avant les modifications. Le DTM Enfant est autorisé à appliquer les modifications uniquement en cas de validation satisfaisante.

6.5 Redondance des esclaves

6.5.1 Vue d'ensemble de la redondance

La duplication des composants critiques d'un système dans le but d'augmenter la fiabilité du système, généralement dans le cas d'une sauvegarde ou d'une sécurité intégrée, est appelée redondance. La Figure 37 représente les scénarios de redondance les plus courants dans le secteur de l'automatisation.

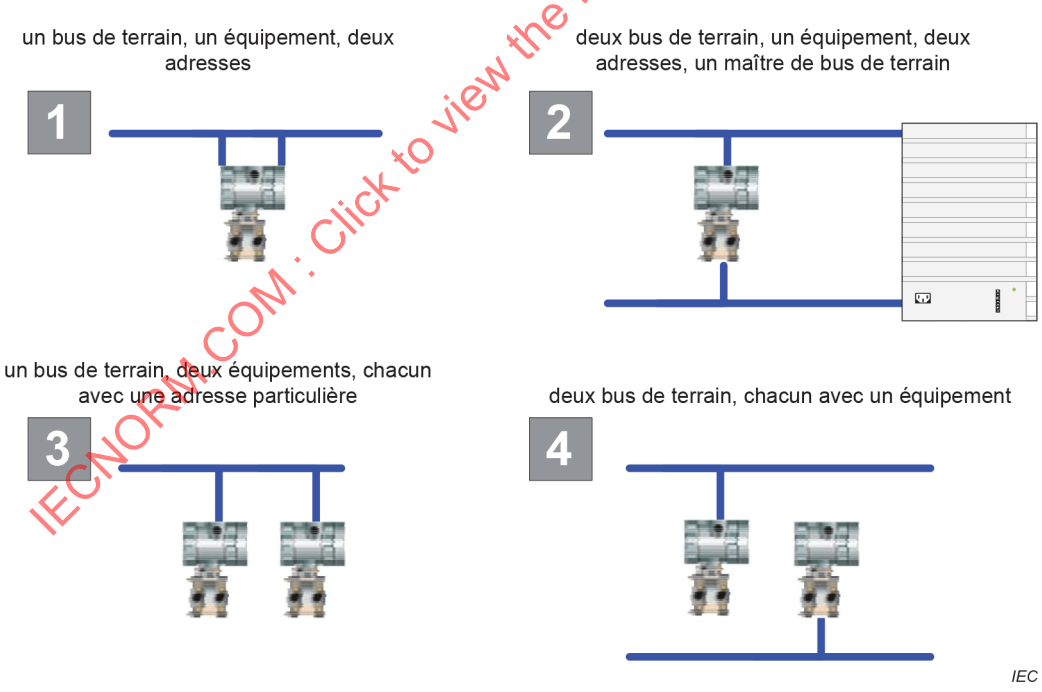


Figure 37 – Scénarios de redondance

La redondance des esclaves au sein des FDT s'applique à un DTM pour la configuration d'un équipement connecté selon l'un des scénarios suivants:

- scénario 1: un bus de terrain qui utilise deux adresses différentes;
- scénario 2: connecté à deux bus de terrain différents contrôlés par un seul maître de bus.

Les équipements redondants séparés (scénarios 3 et 4) ne peuvent pas être gérés au sein des FDT. Ces scénarios nécessitent une fonctionnalité supplémentaire de l'Application-Cadre, par exemple en ce qui concerne la synchronisation des données.

Dans les scénarios 1 et 2, les bus de terrain redondants doivent être gérés par un composant parent sensible à la redondance (par exemple, un DTM de Communication) spécifique à la technologie appropriée des bus de terrain redondants. En règle générale, il convient que ce composant parent soit également capable de gérer en parallèle les DTM pour les dispositifs de terrain redondants et non redondants.

6.5.2 Prise en charge de la redondance dans l'Application-Cadre

Les DTM capables de gérer la redondance des esclaves peuvent être utilisés par toute Application-Cadre du FDT sans nécessiter de prise en charge ou de connaissance particulière de la redondance. Une Application-Cadre du FDT peut de manière facultative prendre en charge les services OnAddedRedundantChild (voir 7.7.4.1) et OnRemovedRedundantChild (voir 7.7.4.2) pour être informée et pouvoir afficher des esclaves redondants dans sa vue topologique, par exemple, un dispositif de terrain connecté à deux lignes différentes ou en utilisant des icônes spécifiques pour les équipements dans une vue topologique.

Au sein d'une Application-Cadre qui n'est pas sensible à la redondance, un DTM qui représente un esclave redondant ne peut pas être perçu au sein de la vue topologique. Cependant, le DTM et le composant parent affichent généralement des informations supplémentaires, par exemple, des adresses supplémentaires de bus de terrain. Néanmoins, au sein d'une telle Application-Cadre, ces DTM fournissent une fonctionnalité totale concernant la redondance des bus de terrain.

Étant donné qu'un esclave redondant est représenté par une instance du DTM, aucune fonctionnalité supplémentaire concernant la synchronisation des ensembles de données ou des données multiutilisateurs n'est exigée.

6.5.3 Composant parent pour le bus de terrain redondant

La communication du bus de terrain pour les esclaves redondants est gérée par un bus de terrain et un composant parent spécifique à la technologie de redondance, par exemple un DTM de Communication spécifique. Tous les bus de terrain utilisés pour connecter des esclaves redondants doivent être gérés par une instance d'un composant parent de ce type. Dans le scénario 2 par exemple, chaque ligne de bus de terrain est représentée par une Voie de Communication appropriée du composant parent. Par conséquent, une Application-Cadre est capable d'utiliser un tel composant parent sans connaître la structure physique du bus de terrain redondant.

Lors de l'exécution du service ValidateAddChild (voir 7.6.2.3) et du service ChildAdded (voir 7.6.2.2), le composant parent est capable de détecter si un DTM d'Équipement à ajouter est capable de gérer la redondance en examinant les informations retournées par le service NetworkManagementInfoRead (voir 7.2.11.1) du DTM instancié. Dans ce cas, une boîte de dialogue spécifique de sélection de l'adresse au sein du composant parent peut être utilisée. Cette sélection de l'adresse est spécifique au bus de terrain et à la redondance. Le composant parent décide de la gestion automatique de la redondance, ou de sa gestion seulement après l'interaction avec l'utilisateur.

Si l'Application-Cadre est sensible à la redondance, le composant parent doit informer l'Application-Cadre de l'existence du DTM redondant par le biais du service OnRemovedRedundantChild (voir 7.7.4.2).

Le composant parent doit être capable de gérer tous les chemins de communication possibles jusqu'à l'équipement redondant. Dans le cadre d'un appel au service NetworkManagementInfoWrite (voir 7.2.11.2), les informations complètes relatives à l'adresse redondante peuvent être fournies à l'instance du DTM qui gère cet équipement redondant. Ce DTM est ensuite capable d'utiliser ces informations pour afficher tous les chemins de communication disponibles. Lors de l'exécution d'un service Connect (voir 7.6.1.2), le DTM fournit ces informations complètes de l'adresse au composant parent. Le composant parent est ensuite capable de sélectionner le chemin de communication actuellement actif. Le chemin de communication peut être modifié au sein du composant parent sans nécessiter d'interaction avec le DTM d'Équipement.

Un composant parent peut également être capable de gérer un DTM représentant un dispositif de terrain non redondant. Dans ce cas, ce DTM est géré sans utiliser les définitions des données spécifiques à la redondance dans les FDT.

6.5.4 Prise en charge de la redondance dans le DTM d'Équipement

Un DTM capable de gérer un équipement redondant fournit des informations supplémentaires au service NetworkManagementInfoRead (voir 7.2.11.1).

Après un appel au service ChildAdded (voir 7.6.2.2), les informations complètes relatives à l'adresse redondante peuvent être fournies par le composant parent au DTM d'Équipement. Le DTM d'Équipement est ensuite capable de détecter si son esclave approprié est utilisé en tant qu'esclave redondant. Ces informations complètes de l'adresse doivent être utilisées par le DTM d'Équipement dans le cadre du service Connect (voir 7.6.1.2), mais peuvent également être utilisées pour les informations relatives au statut et au diagnostic. En général, une gestion supplémentaire de la redondance est exigée au sein du DTM d'Équipement lui-même.

Un DTM qui gère un équipement redondant doit vérifier toutes les adresses fournies par un appel au service NetworkManagementInfoWrite (voir 7.2.11.2). Si le DTM d'Équipement n'est pas capable de gérer ces adresses, par exemple, parce que seul le décalage spécifique à un fournisseur peut être utilisé, il convient de générer un message d'erreur et de retourner une réponse négative au composant parent appelant. Dans ce cas, le composant parent est capable de détecter les DTM d'Équipement qui ne peuvent pas être utilisés au niveau du bus de terrain redondant. Un DTM d'Équipement doit sauvegarder toutes les informations relatives à la redondance dans ses données d'instance.

Un DTM qui représente un esclave redondant peut être utilisé comme enfant d'un composant parent sensible à l'absence de redondance. Dans ce cas, le DTM d'Équipement ne doit pas utiliser la fonctionnalité supplémentaire de redondance du FDT, les interfaces utilisateur graphiques ou les définitions des données spécifiques redondantes dans les FDT.

6.5.5 Balayage et esclaves redondants

Le balayage du bus de terrain fournit une entrée pour chaque adresse d'un esclave redondant. Cette disposition signifie qu'un esclave redondant ne peut pas être détecté.

Dans le scénario 1, le mapping des adresses redondantes avec une instance n'est possible que pour un DTM proprement dit. Dans le scénario 2, le mapping peut s'effectuer uniquement sur la base des connaissances relatives au projet. En général, un balayage n'est pas effectué sur un système redondant.

7 Spécification des services du FDT

7.1 Vue d'ensemble de la spécification des services

L'Article 7 spécifie les services définis pour les objets de FDT. Les définitions des services sont abstraites et ne représentent pas de spécification pour la mise en œuvre. Le mapping entre les descriptions abstraites et les mises en œuvre réelles issues de ces services sont définis dans les parties relatives à la mise en œuvre spécifiques à la technologie.

Les services pour chacun des objets sont organisés en ensembles de services. Chaque ensemble de services définit un ensemble de services associés. L'organisation en ensembles de services est un regroupement logique utilisé dans la présente spécification et un regroupement différent peut être utilisé dans la mise en œuvre.

Chaque service est décrit avec

- un résumé du service;
- un tableau comportant les valeurs des demandes et des réponses;
- une description du service;
- un relevé de disponibilité d'état.

Le résumé du service donne une vue d'ensemble succincte.

Les arguments utilisés pour le service sont décrits dans le tableau qui comporte les valeurs des demandes et des réponses. Pour chaque argument (voir l'Annexe A pour une description des types de données des arguments), il est stipulé si celui-ci est obligatoire ('M' – Mandatory) ou facultatif ('O' – Optional), utilisé pour l'appel au service (Demande) ou fourni comme valeur de retour (Réponse) ou non nécessaire ('-').

La description du service définit le comportement et l'utilisation du service.

Le relevé de disponibilité d'état définit pour chaque service si celui-ci est disponible ou non dans l'état 'configuré' ou 'communication admise'.

Le présent document ne définit pas si la mise en œuvre des services définis est obligatoire ou facultative. Cette définition est fournie par les parties relatives à la mise en œuvre spécifiques à la technologie. Chaque partie spécifique à la technologie fournit une annexe décrivant la mise en œuvre réelle des services (par exemple, le mapping des services avec les méthodes d'interface).

NOTE Le mécanisme d'information qui concerne les services qui ne sont pas mis en œuvre peut être spécifique à la technologie. Pour certaines technologies, le service qui n'est pas mis en œuvre peut être reconnu par le client si un serveur ne fournit pas certaines interfaces. Pour d'autres technologies, les services qui ne sont pas mis en œuvre peuvent être reconnus par des réponses spéciales du service.

L'exécution des services dans un modèle synchrone ou asynchrone dépend de la technologie de mise en œuvre. 'Synchrone' signifie que l'appelant du service est bloqué pendant toute la durée d'exécution du service. 'Asynchrone' signifie que l'appelant est capable de poursuivre les autres opérations jusqu'à l'exécution complète du service. Le modèle d'exécution du service utilisé est défini dans le document relatif au profil d'intégration des modèles d'ojets qui décrit le mapping avec certaines technologies de mise en œuvre. Les deux modèles d'exécution peuvent être utilisés ensemble pour la mise en œuvre des services. De plus, l'annulation de l'exécution active d'un service peut être définie.

7.2 Services du DTM

7.2.1 Services généraux

7.2.1.1 Service PrivateDialogEnabled

Ce service est utilisé par l'Application-Cadre pour signifier à un DTM s'il est autorisé ou non à ouvrir une fenêtre de dialogue privée. Le Tableau 11 présente les arguments du service.

Tableau 11 – Arguments pour le service PrivateDialogEnabled

	Demande	Réponse
Enabled (Boolean)	M	–

Cette condition spéciale est fixée par une Application-Cadre lors de l'exécution. Elle peut s'avérer nécessaire par exemple si

- une action de l'utilisateur en cours ne doit pas être interrompue;
- il n'est pas permis qu'une fenêtre dynamiquement ouverte ait la priorité ou cache d'autres fenêtres, ou
- le DTM est lancé sur un hôte autre que l'interface utilisateur graphique (IUG).

"Enabled" réglé sur FALSE signifie que le DTM ne doit pas ouvrir d'IUG autre que les IUG ouvertes par les services de l'Application-Cadre (par exemple, le service OpenPresentationRequest et le service UserDialog). Si une erreur se produit dans cette condition, le DTM est encore capable d'envoyer une notification à l'Application-Cadre par le biais du service OnErrorMessage (voir 7.7.2.1).

Si le DTM requiert une action de l'utilisateur et doit donc ouvrir une boîte de dialogue avec l'utilisateur, il convient qu'il appelle le service UserDialog de l'Application-Cadre (voir 7.7.8.3). Si aucune IUG ne peut être affichée, le comportement par défaut lors de cette demande consiste à ne pas ouvrir la boîte de dialogue et à fournir au DTM une valeur de retour par défaut. Il incombe donc à l'Application-Cadre de permettre l'ouverture de la boîte de dialogue, de retarder la demande jusqu'à ce que l'action en cours de l'utilisateur soit terminée, ou de refuser la demande par un envoi de la réponse par défaut.

Il convient que les DTM informent l'utilisateur par le biais du service UserDialog si une fonctionnalité spécifique ne peut pas être effectuée parce que les boîtes de dialogue privées ne sont pas admises.

Exemples de boîtes de dialogue privées:

- les boîtes de message (c'est-à-dire la boîte de message normalisée);
- les boîtes de dialogue de sélection de fichier ou d'imprimante (c'est-à-dire fournies par le système d'exploitation);
- les navigateurs web (par défaut);
- les clients de messagerie (par défaut);
- la vue du fichier d'aide;
- la visionneuse manuelle;
- les fenêtres d'attente;
- les applications autonomes externes;
- toute autre fenêtre.

Si l'ouverture des boîtes de dialogue s'effectue sous le contrôle de l'Application-Cadre, celles-ci sont définies comme n'étant pas des boîtes de dialogue privées.

Exemples:

- les boîtes de dialogue ouvertes par le service UserDialog;
- l'interface utilisateur graphique de DTM ouverte par le service OpenPresentationRequest;
- les applications lancées par le service OpenPresentation;
- les fenêtres ouvertes par l'Application-Cadre en raison d'une entrée <Document> dans la réponse reçue du service GetFunctions.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.1.2 Service SetLanguage

Ce service est utilisé par l'Application-Cadre pour informer un DTM lors de l'initialisation de la langue à utiliser dans tous les objets Présentation ainsi que pour les messages à l'utilisateur ou les messages d'erreur. Le Tableau 12 présente les arguments du service.

Tableau 12 – Arguments pour le service SetLanguage

	Demande	Réponse
LanguageID	M	-

L'Application-Cadre configure la langue lors de l'initialisation du DTM, afin que la même langue soit utilisée pour tous les objets Présentation de la même instance. De même, les messages portant sur le service des événements comme le service OnErrorMessage (voir 7.7.2.1) et les contenus interprétables par l'utilisateur retournés par les services GetTypeInformation (voir 7.2.3.1) ou GetDocumentation (voir 7.2.7) doivent utiliser la langue demandée. Si un DTM ne prend pas en charge la langue demandée, il doit utiliser la langue courante (si elle a déjà été initialisée) ou régler la langue courante sur sa langue par défaut lors de la première initialisation.

Les langues d'un DTM prises en charge sont énumérées dans les informations retournées par le service GetTypeInformation (voir 7.2.3.1). Une instance de DTM est toujours initialisée avec une langue. Il revient à un DTM de pouvoir ou non modifier la langue courante d'un objet Présentation lors de son affichage. Le format de sortie pour les nombres, les devises, les durées et les dates doit reposer sur les options régionales du système d'exploitation.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.1.3 Service SetSystemGuiLabel

Ce service est appelé par l'Application-Cadre pour définir un identificateur unique interprétable par l'utilisateur de l'instance d'un DTM, dans le contexte de l'Application-Cadre. Le Tableau 13 présente les arguments du service.

Tableau 13 – Arguments pour le service SetSystemGuiLabel

	Demande	Réponse
windowTitle	M	-

Ce service est appelé par l'Application-Cadre afin de définir une étiquette de système, par exemple pour une boîte de message ou un objet Présentation, faisant partie du DTM et qui ne doit pas être incorporé(e) au sein de l'interface utilisateur de l'Application-Cadre. L'Application-Cadre doit définir l'identificateur unique interprétable par l'utilisateur de l'instance d'un DTM dans le contexte de l'Application-Cadre qui garantit une identification unique entre l'équipement et, par exemple, une boîte de message d'un DTM. Le DTM doit utiliser cette étiquette de système pour tous les types de fenêtres ouvertes par le DTM. Le DTM peut agrandir ce titre avec des informations spécifiques. L'Application-Cadre doit appeler ce service le plus tôt possible. Tant que le service n'a pas été appelé, le DTM doit utiliser le marqueur de l'équipement en tant que chaîne interprétable par l'utilisateur par défaut.

L'identificateur interprétable par l'utilisateur ne doit pas être stocké par le DTM. Il est obligatoire de mettre à jour les étiquettes de toutes les fenêtres ouvertes lorsque ce service est appelé.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.2 Services du DTM relatifs à l'installation

Il convient que chaque DTM qui doit être enregistré au sein du système soit visible pour l'intégration au sein de l'Application-Cadre. En plus de l'enregistrement, le DTM doit indiquer s'il représente un équipement, un module ou un bloc.

Les DTM doivent informer le système lorsqu'ils sont:

- installés;
- désinstallés;
- modifiés (par exemple, de nouveaux types d'équipement sont pris en charge ou le DTM a été mis à jour (correction d'erreur)).

Cette information permet à une Application-Cadre de générer une base de données avec tous les DTM disponibles sur le système. La mise en œuvre détaillée dépend de la technologie utilisée et est décrite dans un document relatif au profil d'intégration des modèles d'ojets.

7.2.3 Services du DTM relatifs aux informations de DTM/équipement

7.2.3.1 Service GetTypeInformation

Ce service retourne des informations générales sur le type de DTM telles que le nom, la version, le fournisseur et les Types d'Équipements pris en charge par le DTM. Le Tableau 14 présente les arguments du service.

Tableau 14 – Arguments pour le service GetTypeInformation (pour DTM)

	Demande	Réponse
fdt:VersionInformation	–	M
fdt:Version	–	M
fdt:DtmDeviceTypes	–	M

Ce service fournit un accès aux informations spécifiques au DTM. Ces données sont disponibles dès que le DTM a été instancié. Le DTM ne nécessite pas de données spécifiques aux instances pour fournir ces informations.

En général, l'Application-Cadre utilise ces informations pour créer des bibliothèques, choisir un DTM ou pour de simples validations.

Si le service est appelé au niveau d'un BTM, les paramètres suivants de demande et de réponse du service sont alors utilisés (voir le Tableau 15).

Tableau 15 – Arguments pour le service GetTypeInformation (pour BTM)

	Demande	Réponse
fdt:VersionInformation	–	M
fdt:Version	–	M
btm:BtmBlockTypes	–	M

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.3.2 Service GetIdentificationInformation

Ce service demande des informations d'identification de l'équipement pour un Type d'Équipement de DTM et un protocole spécifiques. Le Tableau 16 présente les arguments du service.

Tableau 16 – Arguments pour le service GetIdentificationInformation (pour DTM)

	Demande	Réponse
fdt:DtmDeviceType	M	–
fdt:ProtocolId	M	–
fieldbus:DeviceTypeIdentifications ou devIdent:DeviceTypeIdentificationList	–	M

Ce service retourne des informations d'identification de l'équipement pour un type d'équipement ou de bloc demandé. La réponse peut également contenir des informations spécifiques à un protocole.

Si le service est appelé au niveau d'un DTM de Communication, la réponse contient alors la liste devIdent:DeviceTypeIdentification.

Si le service est appelé au niveau d'un BTM, les paramètres suivants de demande et de réponse du service sont alors utilisés (voir le Tableau 17).

Tableau 17 – Arguments pour le service GetIdentificationInformation (pour BTM)

	Demande	Réponse
btm:BtmBlockType	M	–
fdt:ProtocolId	M	–
fieldbus: DeviceTypeIdentifications	–	M

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.3.3 Service HardwareInformation

Ce service demande un balayage pour vérifier la disponibilité du matériel et les informations d'identification correspondantes. Le Tableau 18 présente les arguments du service.

Tableau 18 – Arguments pour le service Hardware information (pour DTM)

	Demande	Réponse
ScanIdentificationList	–	M
ScanIdentificationList peut être soit ident:ScanIdentificationList, soit fieldbus:ScanIdentifications.		

L'Application-Cadre exécute cette fonction pour vérifier si le matériel correspondant est réceptif et pour demander des informations d'identification.

Ce service est disponible dans les états:
[_] 'configuré';
[X] 'communication admise'.

7.2.3.4 Service GetActiveTypeInfo

Le DTM doit fournir des informations relatives au Type d'Équipement du DTM sélectionné. Le Tableau 19 présente les arguments du service.

Tableau 19 – Arguments pour le service GetActiveTypeInfo

	Demande	Réponse
fdt:DtmDeviceType	–	M
net:DtmDeviceInstanceTopology	–	O

Le service doit toujours retourner le Type d'Équipement du DTM actuel, ce qui signifie qu'en cas de mise à jour du DTM, le Type d'Équipement du DTM modifié doit être retourné (par exemple, la version du logiciel la plus récente) à la place du Type d'Équipement du DTM donné lors de l'appel au service Initialize (voir 7.2.4.1).

Le service retourne des informations sur la structure interne de l'équipement correspondant (facultatif). Les informations retournées sont spécifiques à un protocole et à l'équipement.

Si le service est appelé au niveau d'un BTM, les paramètres suivants de demande et de réponse du service sont alors utilisés (voir le Tableau 20).

Tableau 20 – Arguments pour le service GetActiveTypeInfo (pour BTM)

	Demande	Réponse
btm:BtmBlockType		M -

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.4 Services du DTM relatifs au diagramme d'états du DTM

7.2.4.1 Service Initialize

Le service initialise le DTM afin de le passer à un état de fonctionnement. Le Tableau 21 présente les arguments du service.

Tableau 21 – Arguments pour le service Initialize (pour DTM)

	Demande	Réponse
fdt:DtmDeviceType	M	–
user:FDTUserInformation	M	–
fdt:FrameVersion	O	–
fdt:SystemTag	M	–
fdt:FrameObjectReference	M	–
fdt:serviceSucceeded	–	M

Le DTM reçoit les informations relatives au type d'équipement qu'il représente, à l'utilisateur qui accède au DTM, à l'environnement d'exécution, à l'identificateur du DTM et à une référence à l'Application-Cadre proprement dite.

En général, il est prévu qu'un DTM adapte la fonctionnalité fournie selon le rôle de l'utilisateur actuel.

De plus, il est prévu que le DTM adapte son comportement selon la fonctionnalité du FDT de l'environnement d'exécution (selon FrameVersion).

Si le service est appelé au niveau d'un BTM, les paramètres suivants de demande et de réponse du service sont alors utilisés (voir le Tableau 22).

Tableau 22 – Arguments pour le service Initialize (pour BTM)

	Demande	Réponse
btm:BtmBlockType	M	–
user:FDTUserInformation	M	–
fdt:FrameVersion	O	–
fdt:SystemTag	M	–
fdt:FrameObjectReference	M	–
fdt:serviceSucceeded	–	M

Ce service est disponible dans les états:
[X] état initial
[_] 'configuré';
[_] 'communication admise'.

L'appel à ce service aboutit à une transition à l'état 'configuré'.

7.2.4.2 Service SetLinkedCommunicationChannel

Le service informe le DTM de l'existence de la Voie de Communication reliée au sein de la topologie des FDT qui doit être utilisée pour communiquer avec son équipement. Le Tableau 23 présente les arguments du service.

Tableau 23 – Arguments pour le service SetLinkedCommunicationChannel

	Demande	Réponse
fdt:CommunicationObjectReference	M	–
fdt:serviceSucceeded	–	M

La Voie de Communication est définie par l'Application-Cadre: le DTM est capable de vérifier les protocoles de communication pris en charge en appelant le service GetSupportedProtocols (voir 7.6.1.1) et l'information GatewayBusCategory en appelant le service ReadChannelData (voir 7.5.1). En général, l'Application-Cadre est chargée d'établir une liaison valable entre une Voie de Communication et un DTM (voir 4.3). Cependant, cette vérification peut s'avérer nécessaire par exemple si un DTM prend en charge plusieurs protocoles et souhaite identifier celui qui doit être utilisé.

Ce service est disponible dans les états:
[X] 'configuré';
[_] 'communication admise'.

7.2.4.3 Service EnableCommunication

Le service autorise ou permet à un DTM de communiquer avec son équipement. Le Tableau 24 présente les arguments du service.

Tableau 24 – Arguments pour le service EnableCommunication

	Demande	Réponse
Boolean	M	–
fdt:serviceSucceeded	–	M

Le service est appelé par l'Application-Cadre avec TRUE pour faire passer le DTM de l'état 'configuré' à l'état 'communication admise'. Par la suite, le DTM est autorisé à établir une liaison de communication avec son équipement en appelant le service Connect (voir 7.6.1.2) de la Voie de Communication.

Le service est appelé avec FALSE afin de ramener le DTM de l'état 'communication admise' à l'état 'configuré'. Si le DTM a accepté l'appel (fdtServiceSucceeded = TRUE), toutes les liaisons de communication établies doivent alors être fermées (voir les services Disconnect (7.6.1.3) ou AbortRequest (7.6.1.4)). Le DTM est autorisé à refuser le service (fdt:serviceSucceeded = FALSE) si les appels au service de communication auxquels il est impossible de mettre fin sont actifs.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.4.4 Service ReleaseLinkedCommunicationChannel

Le service informe le DTM qu'il doit libérer la Voie de Communication définie par le service SetLinkedCommunicationChannel (voir 7.2.4.2). Le Tableau 25 présente les arguments du service.

Tableau 25 – Arguments pour le service ReleaseLinkedCommunicationChannel

	Demande	Réponse
fdt:serviceSucceeded	–	M

Ce service est disponible dans les états:
[X] 'configuré';
[_] 'communication admise'.

7.2.4.5 Service ClearInstanceData

Ce service permet d'informer le DTM qu'il doit nettoyer, par exemple ses fichiers journaux ou ses protocoles. Les données d'instance sont supprimées par l'Application-Cadre. Le Tableau 26 présente les arguments du service.

Tableau 26 – Arguments pour le service ClearInstanceData

	Demande	Réponse
fdt:serviceSucceeded	–	M

Le service permet d'informer un DTM qu'il doit nettoyer, par exemple les fichiers journaux ou les protocoles de sa mémoire privée (voir 4.9). Après cet appel au service, les données d'instance sont supprimées par l'Application-Cadre. L'Application-Cadre est chargée d'assurer les conditions préalables à la suppression. Autrement dit, l'Application-Cadre doit garantir qu'il est mis fin à toutes les instances de DTM lancées qui sont relatives à ces données d'instance.

Ce service est disponible dans les états:
[X] 'configuré';
[_] 'communication admise'.

7.2.4.6 Service Terminate

Ce service informe le DTM qu'il doit libérer ses références à d'autres composants. L'Application-Cadre met fin au DTM. Le Tableau 27 présente les arguments du service.

Tableau 27 – Arguments pour le service Terminate

	Demande	Réponse
fdt:serviceSucceeded	–	M

Le DTM doit libérer toutes les références aux autres composants et doit mettre fin à l'ensemble des fonctions en attente ou en cours. Il doit également fermer toutes les interfaces utilisateur. Il incombe à un DTM de décider s'il convient de stocker ou non les données transitoires.

Ce service est disponible dans les états:
[X] 'configuré';
[_] 'communication admise'.

7.2.5 Services du DTM relatifs aux fonctions

7.2.5.1 Service GetFunctions

Ce service retourne des informations sur les fonctionnalités normalisées (définies par ApplicationID) ou les fonctionnalités supplémentaires (définies par FunctionId) prises en charge par un DTM. Le Tableau 28 présente les arguments du service.

Tableau 28 – Arguments pour le service GetFunctions

	Demande	Réponse
fdt:OperationPhase	M	–
func:Functions	–	M

Ce service fournit des informations relatives aux fonctions du DTM qui permettent à l'Application-Cadre par exemple de créer un menu spécifique au DTM. Ces données sont disponibles dès que le DTM a été instancié. Les informations peuvent toutefois varier si elles sont spécifiques à l'instance.

Un DTM doit informer l'Application-Cadre des modifications de fonctions (par exemple, le nombre, le statut, l'étiquette, etc.) en appelant le service OnFunctionChanged (voir 7.7.2.4),

Les informations relatives au mode d'exécution des fonctions avec l'interface utilisateur doivent être demandées par le service GetGuiInformation (voir 7.2.5.3). Les appels de fonction sans interface utilisateur doivent être émis par le service InvokeFunctions (voir 7.2.5.2).

Le service fournit également un accès aux documents fournis par un DTM, qui peut être affiché par l'Application-Cadre ou par un programme spécial de visionneuse.

Il est prévu qu'un DTM adapte la liste des fonctions fournie selon le rôle de l'utilisateur actuel. Tant que le DTM ne dispose pas d'informations valables concernant l'utilisateur, il convient qu'il se comporte comme s'il était utilisé par un utilisateur disposant des droits les plus limités ("Observateur").

Il est possible que les fonctions qui présentent des ID de module associées (appelées fonctions de module) ne soient pas affichées directement dans le menu contextuel normalisé du nœud du DTM. Elles peuvent apparaître dans un sous-menu associé à un module de DTM ou dans le menu contextuel d'un nœud de module de DTM. Afin d'assurer que les fonctions du module peuvent être appelées par l'utilisateur, il convient que l'Application-Cadre fournisse un sous-menu "Modules" dans le menu contextuel du nœud du DTM ou fournisse des menus contextuels appropriés pour les modules de DTM, voire offre ces fonctions par d'autres moyens.

Le DTM doit garantir que toutes les fonctions du module accessibles (activées et non cachées) sont valables. Cette disposition signifie que l'Application-Cadre n'est pas chargée de faire concorder les ID du module donné avec la liste des modules existants retournée par le service GetActiveTypeInfo (voir 7.2.3.4).

Si le DTM désactive un groupe de fonctions (type de données des Fonctions), il doit également désactiver toutes les fonctions (type de données des Fonctions) en dessous de ce groupe s'il s'agit du comportement prévu. Si le DTM ne désactive pas les sous-fonctions, l'Application-Cadre peut toujours les utiliser.

Si une Application-Cadre ne prend pas en charge les phases d'exploitation ('notSupported'), il convient que le DTM fournisse toutes les fonctions fondées sur l'état actuel et le rôle de l'utilisateur actuel.

Si le DTM souhaite limiter le nombre d'interfaces utilisateur ouvertes, il convient qu'il désactive les fonctions correspondantes. Si une interface utilisateur est fermée, le DTM doit réactiver la fonction.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.5.2 Service InvokeFunctions

Ce service lance une fonction d'un DTM sans interface utilisateur. Le Tableau 29 présente les arguments du service.

Tableau 29 – Arguments pour le service InvokeFunctions

	Demande	Réponse
func:FDTFunctionCall List	M	–

Ce service exécute une fonction du DTM fournie sans interface utilisateur. Les informations relatives aux fonctions prises en charge par un DTM sont retournées par le service GetFunctions (voir 7.2.5.1).

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.5.3 Service GetGuiInformation

Ce service fournit des informations sur le mode de lancement de l'interface utilisateur d'un DTM. Le Tableau 30 présente les arguments du service.

Tableau 30 – Arguments pour le service GetGuiInformation

	Demande	Réponse
func:FDTFunctionCall	M	–
func:IUGInformation	–	M

Ce service retourne des informations permettant à l'Application-Cadre d'instancier l'interface utilisateur. Selon les informations retournées, l'objet Présentation peut être un type d'objet permettant l'intégration de l'Application-Cadre dans l'interface utilisateur graphique (voir 7.7.8.1). Il peut être également un objet qui fonctionne en dehors de l'interface utilisateur graphique de l'Application-Cadre (voir 7.2.5.4).

L'intégration et l'interaction entre ces objets dépendent de la technologie. Ils sont définis dans les documents relatifs au profil d'intégration des modèles d'ojets de la série IEC 62453.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.5.4 Service OpenPresentation

Ce service demande au DTM d'ouvrir une interface utilisateur pour un appel spécifique de la fonction. Le Tableau 31 présente les arguments du service.

Tableau 31 – Arguments pour le service OpenPresentation

	Demande	Réponse
fdt:invokeID	M	
func:FDTFunctionCall	M	
fdt:windowTitle	M	
fdt:DisplayContext	O	
fdt:serviceSucceeded	–	M

Le service FDTFunctionCall demande l'ouverture de l'objet Présentation correspondant. En général, il incombe à l'Application-Cadre de déterminer le FDTFunctionCall transmis et le DTM de choisir le type de présentation.

Ce service doit toujours apporter une interface utilisateur à la fenêtre active, ou au moins un message d'erreur. Les objets Présentation déjà lancés, identifiés par le service InvokeID, doivent apparaître dans la fenêtre active. La demande d'un objet Présentation déjà lancé avec un nouveau service InvokeID doit être refusée par le DTM.

Le service InvokeID est utilisé par une Application-Cadre pour l'association au sein de la demande du service ClosePresentation (voir 7.2.5.5). De plus, il permet à l'Application-Cadre de gérer une liste des interfaces utilisateur ouvertes.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.5.5 Service ClosePresentation

Ce service demande à un DTM de fermer une interface utilisateur identifiée par le service InvokeID. Le Tableau 32 présente les arguments du service.

Tableau 32 – Arguments pour le service ClosePresentation

	Demande	Réponse
fdt:invokeID	M	
fdt:serviceSucceeded	–	M

Le DTM vérifie simplement s'il est capable ou non de fermer l'interface utilisateur. Dans l'affirmative, le service renvoie une réponse positive et le DTM doit lancer sa procédure de fermeture de l'interface utilisateur. Durant cette fermeture, il doit déverrouiller ses données d'instance et libérer la connexion en ligne avec son équipement si nécessaire. Le DTM lui-même n'est pas désactivé.

En cas d'erreurs, le DTM doit fournir de plus amples informations en appelant le service OnErrorMessage (voir 7.7.2.1)

Le service InvokeID est utilisé par une Application-Cadre pour l'association au sein de la demande du service OpenPresentation approprié (voir 7.2.5.4).

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.6 Services du DTM relatifs aux objets Voies (Channels) – Service GetChannels

Ce service retourne les objets Voies d'un DTM. Le Tableau 33 présente les arguments du service.

Tableau 33 – Arguments pour le service GetChannels

	Demande	Réponse
ChannelObjectReference List	–	M

Ce service retourne les voies d'un DTM. Pour des équipements simples, un objet Voie ne fournit que les informations relatives à l'attribution de la voie.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.7 Services du DTM relatifs à la documentation – service GetDocumentation

Ce service fournit la documentation spécifique au DTM pour une fonction spécifique du DTM. Le Tableau 34 présente les arguments du service.

Tableau 34 – Arguments pour le service GetDocumentation

	Demande	Réponse
func:FDTFunctionCall	M	–
doc:Documentation	–	M

Ce service doit retourner les informations pouvant être utilisées directement pour la documentation. Le contenu des informations retournées est défini par l'Id de l'appel à la fonction transmis, disponible par le biais du service GetFunctions (voir 7.2.5.1). Seules les fonctions avec l'attribut Printable (Imprimable) doivent être prises en charge. L'Application-Cadre doit transformer les informations retournées dans un format interprétable par l'utilisateur.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.8 Services du DTM pour accéder aux données d'instance

7.2.8.1 Généralités

Ces services permettent à une Application-Cadre d'accéder aux paramètres de l'équipement. Le DTM fournit sa représentation actuelle en mémoire de ses données d'instance. Les paramètres disponibles dépendent d'un DTM et du type d'équipement et de bus de terrain.

7.2.8.2 Service InstanceDataInformation

Ce service retourne des informations relatives aux paramètres spécifiques à l'équipement. Le Tableau 35 présente les arguments du service.

Tableau 35 – Arguments pour le service InstanceDataInformation

	Demande	Réponse
fdt:DatasetState	–	M
fdt:StorageState	–	M
param: DtmItemInfoList	–	M
fdt:ModifiedInDevice		M

Ce service fournit des informations relatives aux paramètres et à l'état spécifiques à l'équipement. Les informations peuvent comporter des informations relatives aux paramètres de configuration ainsi que des données relatives à la gestion des biens. Elles peuvent également être vides dans le cas d'équipements sans données publiques. Les contenus des informations fournies peuvent également dépendre de la configuration actuelle de l'équipement et des rôles de l'utilisateur.

Plusieurs réponses peuvent être fournies par le DTM en cas de modification de la liste ou du statut des paramètres.

Le service fournit les données transitoires d'un DTM. Cette disposition signifie que si le DTM est actif ou comporte par exemple une interface utilisateur ouverte, le paramètre fourni peut différer des données d'instance stockées réelles. L'état des données reçues est classé.

Les paramètres fournis peuvent également être disponibles sous forme d'objets Voies (fournis par le service ReadChannelData (voir 7.5.1)). La relation entre ces éléments peut être identifiée par l'information 'SemanticId' (voir le Tableau A.4).

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.8.3 Service InstanceDataRead

Le service fournit un accès en lecture aux données d'instance du DTM. Le Tableau 36 présente les arguments du service.

Tableau 36 – Arguments pour le service InstanceDataRead

	Demande	Réponse
param:ItemId list	M	–
param:DtmItem list	–	M

Ce service permet à une Application-Cadre de demander des données à partir des données d'instance du DTM.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.8.4 Service InstanceDataWrite

Le service fournit un accès en écriture aux données d'instance du DTM. Le Tableau 37 présente les arguments du service.

Tableau 37 – Arguments pour le service InstanceDataWrite

	Demande	Réponse
param:DtmItem list	M	M

Ce service permet à l'Application-Cadre de demander à un DTM de saisir les données spécifiées dans ses données d'instance. De plus, le DTM doit appliquer les règles métier afin de conserver la cohérence des données d'instance.

La réponse comporte les données relatives à l'équipement parmi les données saisies avec succès spécifiées par la demande (elle peut différer de la valeur saisie en raison, par exemple, des procédures d'arrondi internes à l'équipement).

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.9 Services du DTM pour évaluer les données d'instance

7.2.9.1 Service Verify (Vérifier)

Ce service valide l'intégralité des données d'instance à l'aide des règles métier internes du DTM. Le Tableau 38 présente les arguments du service.

Tableau 38 – Arguments pour le service Verify

	Demande	Réponse
fdt:serviceSucceeded	–	M

Une Application-Cadre appelle ce service généralement pour garantir un ensemble de données cohérent, par exemple après une charge persistante ou avant de se connecter.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.9.2 Service CompareDataValueSets

Ce service compare les données d'instance d'un DTM externe prenant en charge le même DatasetId avec ses propres données. Le Tableau 39 présente les arguments du service.

Tableau 39 – Arguments pour le service CompareDataValueSets

	Demande	Réponse
fdt:SystemTag	M	–
fdt:serviceSucceeded	–	M

La structure et les valeurs des paramètres pour la configuration, le paramétrage et l'identification interviennent dans la comparaison. Les paramètres des données qui dépendent de l'exécution (par exemple, les heures de fonctionnement) n'interviennent pas dans la comparaison.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.10 Services du DTM pour accéder aux données de l'équipement

7.2.10.1 Généralités

Ces services fournissent un accès en ligne de l'Application-Cadre aux paramètres spécifiques d'un équipement. Le DTM doit être préparé à plusieurs demandes asynchrones en parallèle. Les demandes doivent être traitées dans l'ordre de leur réception.

Le service ne doit pas modifier les données d'instance.

7.2.10.2 Service DeviceDataInformation

Ce service fournit une liste des paramètres et des valeurs de processus disponibles spécifiques à l'équipement. Le Tableau 40 présente les arguments du service.

Tableau 40 – Arguments pour le service DeviceDataInformation

	Demande	Réponse
param:DtmItemInfo list	–	M

Ce service fournit une liste des paramètres et des valeurs de processus disponibles spécifiques à l'équipement. Au sein d'un DTM, cette liste peut contenir des éléments relatifs aux paramètres de configuration, des valeurs de processus, ainsi que des données relatives à la gestion des biens telles que le frappomètre. Dans l'état 'configuré' du DTM, la liste des éléments retournée repose sur les données d'instance actuelles, qui peuvent différer de celles de la configuration de l'équipement. Dans ce cas, les services DeviceDataRead (voir 7.2.10.3) et DeviceDataWrite (voir 7.2.10.4) peuvent échouer.

Le service fournit une liste des éléments qui peuvent être lus ou saisis à partir du/dans le DTM par le biais du service DeviceDataRead ou saisis dans le DTM par le biais du service DeviceDataWrite. L'équipement proprement dit constitue la source de ces données.

Les éléments fournis dans cette liste peuvent également être contenus dans le service de la voie, le service ReadChannelData (voir 7.5.1) ou le service InstanceDataInformation du DTM (voir 7.2.8.2). Les éléments associés peuvent être identifiés par le biais du service 'SemanticId' (voir le Tableau A.4)

Les informations (éléments de données avec permissions d'accès) peuvent dépendre de la configuration actuelle de l'équipement et du rôle actuel de l'utilisateur. Plusieurs réponses peuvent être fournies par le DTM en cas de modification de la liste ou du statut des paramètres.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.10.3 Service DeviceDataRead

Ce service fournit un accès en lecture aux données de l'équipement. Le Tableau 41 présente les arguments du service.

Tableau 41 – Arguments pour le service DeviceDataRead

	Demande	Réponse
param:ItemId list	M	–
param:DtmItem list	–	M

Ce service permet à une Application-Cadre de demander des données d'un équipement. Les informations relatives aux erreurs doivent être transmises à l'Application-Cadre par la réponse associée.

L'exécution de ce service ne doit pas modifier les données des données d'instance.

Le DTM doit toujours accepter la demande. Si la demande ne peut pas être traitée, la raison de l'échec doit être fournie dans la réponse.

Le DTM doit être capable de traiter plus d'une demande à la fois.

Ce service est disponible dans les états:
[_] 'configuré';
[X] 'communication admise'.

7.2.10.4 Service DeviceDataWrite

Ce service fournit un accès en écriture aux données de l'équipement. Le Tableau 42 présente les arguments du service.

Tableau 42 – Arguments pour le service DeviceDataWrite

	Demande	Réponse
param:DtmItem list	M	M

Ce service permet à l'Application-Cadre de demander à un DTM de saisir les données spécifiées dans son équipement, selon les règles spécifiques à l'équipement. Les informations relatives aux erreurs doivent être transmises à l'Application-Cadre par la réponse associée.

L'exécution de ce service ne doit pas modifier les données des données d'instance.

Le DTM doit vérifier s'il peut ou non manipuler le fanion 'ModifiedInDevice' (voir le Tableau A.4) dans les données d'instance en demandant un verrouillage. Si la demande de verrouillage échoue, le DTM doit également refuser l'appel DeviceDataWrite. Une réponse appropriée doit être fournie.

Le DTM doit toujours accepter la demande. Si la demande ne peut pas être traitée, la raison de l'échec doit être fournie de manière asynchrone dans la réponse.

Il convient que le DTM soit capable de traiter plus d'une demande à la fois.

La réponse sous forme de liste DtmItem comporte les données de l'équipement des données saisies avec succès (elle peut différer de la valeur saisie en raison, par exemple, des procédures d'arrondi internes à l'équipement).

Ce service est disponible dans les états:
[_] 'configuré';
[X] 'communication admise'.

7.2.11 Services du DTM relatifs aux informations de gestion de réseau

7.2.11.1 Service NetworkManagementInfoRead

Ce service fournit un accès en lecture aux informations relatives à la gestion de réseau. Le Tableau 43 présente les arguments du service.

Tableau 43 – Arguments pour le service NetworkManagementInfoRead

	Demande	Réponse
net:NetworkInfo	–	M

Ce service fournit un accès en écriture aux informations relatives à la gestion de réseau. Les informations retournées sont spécifiques à un protocole. Elles comportent par exemple l'adresse du bus de l'équipement, le marqueur, ainsi que d'autres paramètres de configuration spécifiques au bus.

L'utilisation de ce service est détaillée en 6.1.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.11.2 Service NetworkManagementInfoWrite

Ce service fournit un accès en écriture aux informations relatives à la gestion de réseau. Le Tableau 44 présente les arguments du service.

Tableau 44 – Arguments pour le service NetworkManagementInfoWrite

	Demande	Réponse
net:NetworkInfo	M	–

Ce service fournit un accès en écriture aux informations de gestion de réseau qui peuvent être lues par le service NetworkManagementInfoRead (voir 7.2.11.1).

L'utilisation de ce service est détaillée en 6.1.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.12 Services du DTM relatifs au fonctionnement en ligne

7.2.12.1 Service DeviceStatus

Ce service fournit des informations relatives au statut de l'équipement. Le Tableau 45 présente les arguments du service.

Tableau 45 – Arguments pour le service DeviceStatus (pour DTM)

	Demande	Réponse
status:DTMDeviceStatus	–	M

Le DTM doit charger le statut actuel de l'équipement. Selon le protocole de bus de terrain, le DTM peut également charger ses informations de diagnostic actuelles. Selon ces informations, le DTM doit fournir le statut dans un format interprétable par l'utilisateur. La fonction ne doit pas ouvrir d'interface utilisateur pour permettre la vérification des réseaux complets.

Un BTM doit retourner le statut du bloc associé.

Ce service est disponible dans les états:
[_] 'configuré';
[X] 'communication admise'.

7.2.12.2 Service CompareDataValueSetWithDeviceData

Ce service compare les données d'instance du DTM avec les données de l'équipement. Le Tableau 46 présente les arguments du service.

Tableau 46 – Arguments pour le service CompareInstanceDataWithDeviceData (pour DTM)

	Demande	Réponse
onlineComp:DTMOnlineCompare	–	M

Ce service est utilisé pour le traitement par lots et doit fonctionner sans interface utilisateur. Il compare ses données d'instance avec les données de l'équipement chargées à partir de celui-ci. Si le DTM est capable de charger les données de l'équipement, il doit alors comparer les données et indiquer dans sa réponse si les données sont égales ou non. Sinon, la réponse doit comporter les informations relatives aux erreurs de communication.

Si le DTM ne comporte pas de données comparables en ligne, il doit retourner la réponse 'noComparableData' en tant que valeur du 'StatusFlag'.

Il convient que la comparaison n'inclue que les données significatives pour la configuration, le paramétrage et l'identification de l'équipement. Il convient de ne pas inclure les données relatives à l'exécution (par exemple, les heures de fonctionnement).

Ce service est disponible dans les états:
[_] 'configuré';
[X] 'communication admise'.

7.2.12.3 Service WriteDataToDevice

Ce service demande à saisir des données en ligne dans l'équipement. Le Tableau 47 présente les arguments du service.

Tableau 47 – Arguments pour le service WriteDataToDevice (pour DTM)

	Demande	Réponse
fdt:serviceSucceeded	–	M

Ce service est initié par l'Application-Cadre. Il est obligatoire pour tous les DTM qui prennent en charge les données en ligne. Le service permet de saisir dans l'équipement tous les paramètres nécessaires à sa mise en service à partir des données d'instance du DTM.

En cas d'erreurs, le DTM doit informer l'Application-Cadre par le biais du service OnErrorMessage (voir 7.7.2.1).

Ce service est disponible dans les états:
[_] 'configuré';
[X] 'communication admise'.

7.2.12.4 Service ReadDataFromDevice

Ce service demande à lire des données en ligne à partir de l'équipement. Le Tableau 48 présente les arguments du service.

Tableau 48 – Arguments pour le service ReadDataFromDevice (pour DTM)

	Demande	Réponse
fdt:serviceSucceeded	–	M

Ce service est initié par l'Application-Cadre. Il est obligatoire pour tous les DTM qui prennent en charge les données en ligne. Le DTM utilise ce service pour lire dans l'équipement tous les paramètres nécessaires à sa mise en service à partir des données d'instance du DTM.

En cas d'erreurs, le DTM doit informer l'Application-Cadre par le biais du service OnErrorMessage (voir 7.7.2.1).

Ce service est disponible dans les états:
[_] 'configuré';
[X] 'communication admise'.

7.2.13 Services du DTM relatifs à la synchronisation des données

7.2.13.1 Service OnLockInstanceData

Dans le cas d'un environnement multiutilisateur, il est nécessaire d'informer le DTM de ses droits d'accès actuels à ses données d'instance, notamment si un autre DTM obtient l'accès en écriture aux données d'instance. Voir 8.5 pour de plus amples informations sur les scénarios dédiés à plusieurs utilisateurs. Ce service informe un DTM du verrouillage des données par une autre de ses instances. Le Tableau 49 présente les arguments du service.

Tableau 49 – Arguments pour le service OnLockInstanceData

	Demande	Réponse
fdt:SystemTag	M	–
user:UserName	M	–

Si un autre DTM a verrouillé les données d'instance par le biais du service LockInstanceData (voir 7.7.6.1), l'Application-Cadre doit envoyer OnLockInstanceData à toutes les instances du DTM qui ont une référence aux mêmes données d'instance.

Lors de la réception de cette notification, un DTM ne doit admettre aucune opération de modification des données relatives à l'instance, par exemple en désactivant ses champs d'entrée dans le cas d'une interface utilisateur ouverte.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.13.2 Service OnUnlockInstanceData

Dans le cas d'un environnement multiutilisateur, il est nécessaire d'informer un DTM de son mode d'accès actuel, notamment si un autre DTM obtient l'accès en lecture/écriture aux données relatives à l'instance. Ce service informe un DTM qu'un autre DTM a libéré le verrouillage des données d'instance. Le Tableau 50 présente les arguments du service.

Tableau 50 – Arguments pour le service OnUnlockInstanceData

	Demande	Réponse
fdt:SystemTag	M	–
user:UserName	M	–
ServiceSucceeded	–	M

Si un DTM a déverrouillé des données d'instance par le biais du service UnlockInstanceData (voir 7.7.6.2), l'Application-Cadre doit envoyer des informations par le biais du service OnUnlockInstanceData à tous les DTM ayant une référence aux mêmes données d'instance. Lors de la réception de cette notification, un DTM qui prend en charge la synchronisation des données retourne une réponse positive et doit permettre l'alternance des données d'instance. Les DTM qui ne prennent pas en charge la synchronisation des données doivent retourner une valeur de réponse négative et ne doivent pas admettre de modification de données. Voir 8.5.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.13.3 Service OnInstanceDataChanged

Dans le cas d'un environnement multiutilisateur, il est nécessaire d'informer le DTM des modifications de données. Ce service informe un DTM des modifications apportées aux données d'instance. Le Tableau 51 présente les arguments du service.

Tableau 51 – Arguments pour le service OnInstanceDataChanged

	Demande	Réponse
fdt:SystemTag	M	–
Information	M	–

Si un DTM comporte des données modifiées et stockées, il doit appeler le service InstanceDataChanged de l'Application-Cadre (voir 7.7.6.3) avec les informations des données modifiées. L'Application-Cadre les transfère à tous les autres DTM qui ont une référence aux mêmes données d'instance et prend en charge le verrouillage synchronisé en appelant le service OnInstanceDataChanged de ce DTM.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.13.4 Service OnChildInstanceDataChanged

Dans le cas d'une topologie des FDT, il est nécessaire d'informer le DTM Parent de l'existence des données modifiées. Ce service informe un DTM des modifications apportées aux données d'instance d'un DTM Enfant. Le Tableau 52 présente les arguments du service.

Tableau 52 – Arguments pour le service OnInstanceChildDataChanged

	Demande	Réponse
fdt:SystemTag	M	–

Si un DTM a modifié une ou plusieurs données, il doit appeler le service InstanceDataChanged de l'Application-Cadre (voir 7.7.6.3) avec les informations qui comportent les modifications spécifiques à l'instance. L'Application-Cadre transfère ce document en appelant le service OnInstanceDataChanged du DTM (voir 7.2.13.3) à partir de tous les DTM qui ont une référence aux mêmes données d'instance.

Concernant les DTM Parents correspondants (parent principal, parents secondaires (voir service GetParentNodes, 7.7.3.5)), l'Application-Cadre doit mettre en œuvre le comportement suivant: l'Application-Cadre doit envoyer une notification au DTM Parent correspondant par le biais du service OnChildInstanceDataChanged de ce DTM (voir 7.2.13.4) au sein d'un environnement multiutilisateur. Cette notification n'est envoyée qu'à un seul DTM Parent: elle doit être envoyée à celui qui comporte le verrouillage des données d'instance associées.

Si aucun DTM Parent n'est actuellement lancé, l'Application-Cadre doit lancer le DTM Parent.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.2.14 Services du DTM relatifs à l'importation et à l'exportation

7.2.14.1 Service Export (Exportation)

Ce service permet de construire une image d'exportation de l'ensemble des données persistantes du DTM. Le Tableau 53 présente les arguments du service.

Tableau 53 – Arguments pour le service Export

	Demande	Réponse
fdt:StorageObject	–	M

Ce service permet de construire une image d'exportation de l'ensemble des données persistantes du DTM indépendamment de leur stockage ou non dans la mémoire de Projet de l'Application-Cadre ou dans une mémoire DTM privée (voir 4.9). Les données retournées par le service doivent inclure les données relatives aux instances du DTM et celles qui ne sont pas relatives aux instances.

Ce service est disponible dans les états:
[X] 'configuré';
[_] 'communication admise'.

7.2.14.2 Service Import (Importation)

Ce service permet de réimporter dans le DTM les données précédemment exportées par le service Export (voir 7.2.14.1). Le Tableau 54 présente les arguments du service.

Tableau 54 – Arguments pour le service Import

	Demande	Réponse
fdt:StorageObject	M	–

Ce service permet de réimporter dans le DTM les données précédemment exportées par le service Export (voir 7.2.14.1).

Ce service est disponible dans les états:
[X] 'configuré';
[_] 'communication admise'.

7.3 Services des objets Présentation (Presentation)

L'interaction et le contrôle des états entre l'Application-Cadre, l'objet Présentation et le DTM dépendent de la technologie et sont définis dans un document relatif au profil d'intégration des modèles d'ojets.

7.4 Service des objets Voies (Channels)

7.4.1 Vue d'ensemble du service des objets Voies

Le paragraphe 7.4 définit les services de base de la voie communs aux Voies de Communication et aux Voies de Processus.

7.4.2 Service ReadChannelInformation

Ce service retourne les informations générales de la voie (indépendantes d'un protocole). Le Tableau 55 présente les arguments du service.

Tableau 55 – Arguments pour le service ReadChannelInformation

	Demande	Réponse
fdt:Tag		M
fdt:ChannelPath		M
fdt:ChannelId		M
fdt:FrameApplicationTag		O

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.4.3 Service WriteChannelInformation

Ce service saisit les informations générales de la voie (indépendantes d'un protocole). Le Tableau 56 présente les arguments du service.

Tableau 56 – Arguments pour le service WriteChannelInformation

	Demande	Réponse
fdt:ProtectedByChannelAssignment	M	–
fdt:FrameApplicationTag	O	–
fdt:serviceSucceeded		M

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.5 Services des objets Voie de processus (Process channels) – services pour les informations relatives aux E/S

7.5.1 Service ReadChannelData

Ce service retourne des informations avec des paramètres de la voie dépendants du bus de terrain. Le Tableau 57 présente les arguments du service.

Tableau 57 – Arguments pour le service ReadChannelData

	Demande	Réponse
fdt:ProtocolId	M	
fieldbus:ChannelData		M

Le service retourne une description des paramètres spécifiques à la voie. Le bus de terrain est sélectionné par le paramètre ProtocolId.

Il convient que l'appelant du service ReadChannelData utilise le ProtocolId du membre NetworkInfo dans l'information DtmDeviceInstanceTopology disponible par le biais du service GetActiveTypeInfo.

Les paramètres retournés sont notamment utilisés pour l'attribution de la voie.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.5.2 Service WriteChannelData

Ce service définit les modifications des paramètres spécifiques à la voie. Le Tableau 58 présente les arguments du service.

Tableau 58 – Arguments pour le service WriteChannelData

	Demande	Réponse
fdt:ProtocolId	M	
fieldbus:ChannelData	M	
fdt:serviceSucceeded		M

Le service reçoit les paramètres spécifiques à la voie selon une description spécifique au bus de terrain. Le bus de terrain est défini par le paramètre ProtocolId.

Le service retourne une réponse indiquant si la voie a accepté toutes les informations avec toutes les modifications ou si la voie a refusé toutes les modifications transférées. Dans ce cas, la voie informe l'Application-Cadre de l'erreur en détail par le biais du service OnErrorMessage (voir 7.7.2.1).

Le service fonctionne à partir des données transitoires d'un DTM. Cette disposition signifie que les nouvelles données ne sont pas stockées implicitement.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.6 Services de l'objet Voie de Communication (Communication Channel)

7.6.1 Services relatifs à la communication

7.6.1.1 Service GetSupportedProtocols

Ce service retourne des informations relatives aux protocoles de la Voie de Communication pris en charge. Le Tableau 59 présente les arguments du service.

Tableau 59 – Arguments pour le service GetSupportedProtocols

	Demande	Réponse
fdt:BusCategory List	–	M

Le service retourne les UUID du protocole du bus de terrain. Les protocoles pris en charge peuvent être statiques ou dépendre de la configuration de l'équipement. Si une voie prend en charge plus d'un protocole durant son exécution, elle doit prendre en charge tous les protocoles en parallèle.

Le service n'est pas autorisé à modifier les protocoles pris en charge si un DTM est relié à la voie, car une modification peut entraîner une topologie non valable (voir 4.3). Les protocoles pris en charge peuvent être déterminés lors de la planification de la topologie. Par conséquent, si la Voie de Communication peut être configurée pour prendre en charge différents protocoles, cette opération n'est permise qu'en l'absence de DTM reliés.

Ce service est disponible dans les états:
[_] 'configuré';
[X] 'communication admise'.

7.6.1.2 Service Connect (Connexion)

Ce service établit une nouvelle liaison de communication avec un équipement. Le Tableau 60 présente les arguments du service.

Tableau 60 – Arguments pour le service Connect

	Demande	Réponse
fdt:CommunicationClientObjectReference	M	–
fdt:SystemTag	O	–
fdt:BusCategory	M	–
fieldbus:ConnectRequest	M	–
fieldbus:ConnectResponse	–	M
fdt:CommunicationReferenceID	–	M

Le service prend les informations avec les paramètres de communication spécifiques au bus de terrain. Le protocole du bus de terrain à utiliser est identifié par le paramètre BusCategory.

La demande comporte d'autres informations spécifiques au protocole du bus de terrain pour la Voie de Communication concernant le mode d'établissement de la connexion, par exemple, les informations telles que le nombre de répétitions ou de préambules, etc. Il relève de l'environnement de décider si ces informations sont adaptées.

De plus, la demande comporte des informations spécifiques au bus de terrain et à un protocole concernant le mode d'adressage de l'équipement connecté à un bus de terrain particulier.

Le SystemTag (voir le Tableau A.4) fourni dans la demande de connexion désigne le SystemTag du client de la communication. Il peut être utilisé pour accéder au DTM en appelant le service GetDtm (voir 7.7.3.7). Si le SystemTag n'est pas fourni, le client de la communication n'est alors pas un DTM (il peut s'agir de l'Application-Cadre ou d'un autre composant).

Ce service est disponible dans les états:
[_] 'configuré';
[X] 'communication admise'.

7.6.1.3 Service Disconnect

Ce service libère une liaison de communication avec un équipement. Dans le cas de deux connexions ou plus à un même équipement, la liaison est identifiée par la référence de communication retournée par le service Connect (voir 7.6.1.2). Le Tableau 61 présente les arguments du service.

Tableau 61 – Arguments pour le service Disconnect

	Demande	Réponse
fdt:CommunicationReferenceID	M	–
fieldbus:DisConnectRequest	M	–
fieldbus:DisConnectResponse	–	M

Ce service permet à la Voie de Communication de transmettre à l'appelant du service les informations indiquant si la connexion est libérée ou non.

Le service entraîne la libération de toutes les données de réponse en attente, ainsi qu'au moins la libération du service CommunicationClientObjectReference transmis par le service Connect (voir 7.6.1.2).

Ce service est disponible dans les états:
[_] 'configuré';
[X] 'communication admise'.

7.6.1.4 Service AbortRequest

Ce service arrête prématurément une liaison de communication avec un équipement sans aucune réponse. Le Tableau 62 présente les arguments du service.

Tableau 62 – Arguments pour le service AbortRequest

	Demande	Réponse
fdt:CommunicationReferenceID	M	–

Le service met fin à toutes les demandes en attente et retourne une réponse sans attendre de résultat. La fin de la connexion n'est pas confirmée.

Ce service est disponible dans les états:
[_] 'configuré';
[X] 'communication admise'.

7.6.1.5 Service de notification AbortIndication

Ce service fournit une notification indiquant qu'une connexion (identifiée par le CommunicationReferenceID) a été arrêtée prématurément. Le Tableau 63 présente les arguments du service.

Tableau 63 – Arguments pour le service AbortIndication

	Demande	Réponse
fdt:CommunicationReferenceID		M

Le service retourne la notification d'arrêt prématuré d'une connexion à un composant de communication connecté ou à un DTM connecté. Il est également mis fin à toutes les demandes en attente sur cette connexion.

La fin de la connexion n'est pas confirmée.

La fin d'une connexion peut résulter d'un service appelé ou peut survenir "spontanément" (par exemple, si l'absence d'un équipement est notée automatiquement par l'infrastructure de communication sous-jacente).

Ce service est disponible dans les états:
[_] 'configuré';
[X] 'communication admise'.

7.6.1.6 Service Transaction

Le service réalise un échange de la structure de données avec un équipement spécifié par le client de communication. Le Tableau 64 présente les arguments du service.

Tableau 64 – Arguments pour le service Transaction

	Demande	Réponse
fdt:CommunicationReferenceID	M	–
fieldbus:TransactionRequest	M	–
fieldbus:TransactionReponse	–	M

Le service retourne des réponses de communication spécifiques à un protocole aux demandes de même nature.

Il convient que les développeurs de Voies de Communication ne prévoient pas qu'il y ait une seule demande en attente à la fois pour un équipement donné. Exemple: plusieurs clients (de l'Application-Cadre et des DTM d'Équipement, par exemple) tentent de récupérer des informations à partir du même équipement.

Même si le protocole de bus de terrain sous-jacent admet l'envoi d'une seule demande à la fois à un ou plusieurs équipements, les Voies de Communication doivent être capables de gérer un certain nombre de demandes.

Il est prévu que les demandes soient traitées par la Voie de Communication dans l'ordre de leur réception, sauf spécification contraire du protocole.

Il convient que les développeurs des DTM d'Équipement tiennent compte du fait que l'infrastructure de communication utilisée génère des retards dans la communication. Il convient donc que les DTM d'Équipement limitent le nombre de demandes de communication.

De même, le DTM d'Équipement doit être capable de gérer une demande refusée, car les raisons expliquant le refus d'une demande de transaction peuvent être nombreuses.

Ce service est disponible dans les états:
[_] 'configuré';
[X] 'communication admise'.

7.6.1.7 Service SequenceDefine

Le service SequenceDefine définit une séquence de transactions de communication et les prépare pour leur exécution. Le Tableau 65 présente les arguments du service. La demande du service SequenceStart signifie la fin des transactions de communication.

Un DTM permet de définir une seule séquence et toutes les transactions demandées par les séquences précédentes sont terminées.

Tableau 65 – Arguments pour le service SequenceDefine

	Demande	Réponse
fdt:CommunicationReferenceID	M	
fieldbus:SequenceDefine	M	–
fdt:serviceSucceeded		M

La Voie de Communication doit maintenir la séquence définie jusqu'à la demande du service SequenceStart.

Ce service est disponible dans les états:
[_] 'configuré';
[X] 'communication admise'.

7.6.1.8 Service SequenceStart

Le service SequenceStart termine l'exécution de la séquence de transactions de communication définies par le service SequenceDefine (voir 7.6.1.7). Le Tableau 66 présente les arguments du service.

Tableau 66 – Arguments pour le service SequenceStart

	Demande	Réponse
fdt:CommunicationReferenceID	M	
fdt:serviceSucceeded		M

La séquence de transactions de communication est terminée dans l'ordre de définition des transactions sans interruption jusqu'à l'exécution du service de la dernière transaction. La Voie de Communication informe le client du résultat de chaque transaction lorsqu'elle est terminée.

Ce service est disponible dans les états:
[_] 'configuré';
[X] 'communication admise'.

7.6.2 Services relatifs à la gestion de la sous-topologie

7.6.2.1 Service ValidateAddChild

Le service confirme si un DTM donné, spécifié par son SystemTag, peut être ajouté ou non à la sous-topologie (voir le 4.3). Le Tableau 67 présente les arguments du service.

Tableau 67 – Arguments pour le service ValidateAddChild

	Demande	Réponse
fdt:SystemTag	M	–
fdt:serviceSucceeded	–	M

La voie valide la liaison. Si la liaison est valable, la voie retourne une réponse positive (succès).

Lorsqu'il est nécessaire que le DTM dispose de plus d'informations concernant le DTM Enfant, il est capable d'accéder au DTM par le biais du service GetDtm (voir 7.7.3.7) en transmettant le SystemTag reçu.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.6.2.2 Service ChildAdded

La voie est informée qu'un nouveau DTM a été ajouté à la sous-topologie. Le Tableau 68 présente les arguments du service.

Tableau 68 – Arguments pour le service ChildAdded

	Demande	Réponse
fdt:SystemTag	M	–

La voie est informée qu'un nouveau DTM, spécifié par son SystemTag, a été ajouté à la sous-topologie.

Lorsqu'il est nécessaire que la voie dispose de plus d'informations concernant le DTM Enfant, elle est capable d'accéder au DTM par le biais du service GetDtm (voir 7.7.3.7) en transmettant le SystemTag reçu.

Ce service ne doit être utilisé que pour indiquer une modification de la topologie. En cas de rechargement, par exemple de données relatives au projet, ce service ne doit pas être appelé.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.6.2.3 Service ValidateRemoveChild

Ce service confirme si un DTM Enfant donné, spécifié par son SystemTag, peut être retiré de la sous-topologie. Le Tableau 69 présente les arguments du service.

Tableau 69 – Arguments pour le service ValidateRemoveChild

	Demande	Réponse
fdt:SystemTag	M	–
fdt:serviceSucceeded	–	M

La voie doit confirmer si le DTM peut être retiré de la topologie.

Lorsqu'il est nécessaire que le DTM dispose de plus d'informations concernant le DTM Enfant, il est capable d'accéder au DTM par le biais du service GetDtm (voir 7.7.3.7) en transmettant le SystemTag reçu.

La validation doit inclure une vérification de la connexion active et retourner une réponse négative (échec) si la connexion est active par le biais de cette voie.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.6.2.4 Service ChildRemoved

Ce service permet d'informer la voie du retrait d'un DTM relié de la sous-topologie. Le Tableau 70 présente les arguments du service.

Tableau 70 – Arguments pour le service ChildRemoved

	Demande	Réponse
fdt:SystemTag	M	–

La voie est informée du retrait d'un DTM relié, spécifié par son SystemTag, de la sous-topologie.

Lorsqu'il est nécessaire que la voie dispose de plus d'informations concernant le DTM Enfant, elle est capable d'accéder au DTM par le biais du service GetDtm (voir 7.7.3.7) en transmettant le SystemTag reçu.

Ce service ne doit être utilisé que pour indiquer une modification de la topologie. Dans le cas du déchargement, par exemple, de données relatives au projet, ce service ne doit pas être appelé.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.6.2.5 Service SetChildrenAddresses

Ce service demande le réglage d'une adresse de bus pour une liste spécifiée d'équipements et retourne la liste d'équipements avec les informations de réussite (voir 6.1). Le Tableau 71 présente les arguments du service.

Tableau 71 – Arguments pour le service SetChildrenAddresses

	Demande	Réponse
devList DeviceList	M	–
devList DeviceList	–	M

Ce service demande le réglage de l'adresse du bus pour un équipement ou une liste d'équipements. La demande peut préciser que la Voie de Communication appelée doit ouvrir une interface utilisateur pour demander les réglages des adresses d'équipement à l'utilisateur. Pour obtenir une réponse agréée, les informations relatives aux erreurs sont incluses dans les informations retournées.

Lors de l'exécution de ce service, le service OpenPresentationRequest (7.7.8.1) peut être utilisé pour demander une sélection d'adresses à l'utilisateur.

Le service NetworkManagementInfoWrite permet de régler l'adresse du bus sur un DTM (voir 7.2.11.2). Ce DTM ne doit pas utiliser ces informations pour exécuter les transactions de communication, qui définissent l'adresse dans le dispositif de terrain.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.6.3 Services relatifs à l'interface utilisateur graphique (IUG) et aux fonctions

7.6.3.1 Service GetChannelFunctions

Ce service retourne les informations relatives aux fonctions d'un objet Voie du DTM. Le Tableau 72 présente les arguments du service.

Tableau 72 – Arguments pour le service GetChannelFunctions

	Demande	Réponse
fdt:OperationPhase	M	–
func:Functions	–	M

Ce service fournit des informations sur les fonctions fournies par la voie.

Une voie doit informer l'Application-Cadre des modifications de fonctions (par exemple, le numéro, le statut, l'étiquette, etc.) en appelant le service OnFunctionChanged (voir 7.7.2.4).

Plusieurs réponses peuvent être fournies. L'Application-Cadre utilise généralement ces informations pour créer des menus.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.6.3.2 Service GetGuiInformation

Ce service fournit des informations concernant le mode de lancement de l'interface utilisateur d'une voie. Le Tableau 73 présente les arguments du service.

Tableau 73 – Arguments pour le service GetGuiInformation

	Demande	Réponse
func:FDTFunctionCall	M	–
func:IUGInformation	–	M

Ce service retourne des informations qui permettent à l'Application-Cadre d'instancier l'interface utilisateur. Les voies ne doivent prendre en charge que les objets Présentation permettant l'intégration de l'Application-Cadre dans l'interface utilisateur graphique.

L'intégration et l'interaction entre ces objets dépendent de la technologie. Ils sont définis dans les documents relatifs au profil d'intégration des modèles d'ojets.

Ce service est disponible dans les états:
[X] 'configuré';
[X] 'communication admise'.

7.6.4 Service Scan

Ce service réalise le balayage de la sous-topologie (voir 6.2.2). Le Tableau 74 présente les arguments du service.

Tableau 74 – Arguments pour le service Scan

	Demande	Réponse
devList:BusAddressRange	O	–
fieldbus:ScanIdentifications	–	M

Ce service retourne des informations qui comportent une liste d'informations relatives au bus de terrain pour identifier les équipements connectés. L'objet BusAddressRange facultatif transmis peut limiter la plage de balayage à des étendues ou des adresses spécifiques de l'équipement.

Ce service est disponible dans les états:
[_] 'configuré';
[X] 'communication admise'.

7.7 Services de l'Application-Cadre

7.7.1 Disponibilité générale des états

Tous les services de l'Application-Cadre sont disponibles dans les états du DTM:
[X] 'configuré';
[X] 'communication admise'.

7.7.2 Services de l'Application-Cadre relatifs aux événements généraux

7.7.2.1 Service OnErrorMessage

Ce service permet à une Application-Cadre de recevoir une notification d'un DTM concernant les erreurs survenues lors d'un appel à un service. Le Tableau 75 présente les arguments du service.

Tableau 75 – Arguments pour le service OnErrorMessage

	Demande	Réponse
fdt:SystemTag	M	–
fdt:ErrorMessage	M	–

Le service est nécessaire si le DTM fonctionne sans interface utilisateur. Si un DTM fonctionne sans interface utilisateur, il n'est pas autorisé à afficher des messages d'erreur dans ses propres fenêtres de dialogue.

Il incombe à l'Application-Cadre de gérer les informations. En cas d'erreurs ou d'avertissements, la chaîne interprétable par l'utilisateur peut s'afficher dans une boîte de dialogue de l'Application-Cadre.

Afin de fournir aux utilisateurs des informations utiles (par exemple, relatives aux erreurs), il est recommandé d'utiliser ce service en cas d'échec d'un appel à un service et de non-utilisation du service UserDialog (voir 7.7.8.3).

7.7.2.2 Service OnProgress

Ce service permet à une Application-Cadre de recevoir une notification d'un DTM concernant la progression du traitement d'un appel à un service. Le Tableau 76 présente les arguments du service.

Tableau 76 – Arguments pour le service OnProgress

	Demande	Réponse
fdt:SystemTag	M	–
fdt:ProgressInformation	M	–

Les DTM doivent utiliser ce service lors des appels aux services actifs, de plus longue durée, pour informer l'Application-Cadre et au moins l'utilisateur des activités en cours. Si un DTM est incapable de déterminer la progression réelle, il peut s'avérer utile de modifier, par exemple, le titre.

Le mode de traitement des informations relève de l'Application-Cadre.

7.7.2.3 Service OnOnlineStatusChanged

Ce service permet à une Application-Cadre de recevoir une notification d'un DTM concernant le statut de la liaison de communication. Le Tableau 77 présente les arguments du service.

Tableau 77 – Arguments pour le service OnOnlineStatusChanged

	Demande	Réponse
fdt:SystemTag	M	–
fdt:OnlineStatus (previous)	M	–
fdt:OnlineStatus (current)	M	
fdt:CommunicationError	O	

Les DTM doivent utiliser ce service pour informer l'Application-Cadre des modifications du statut de la liaison de communication. Le DTM doit spécifier le nouveau et le précédent statut en ligne ainsi que d'autres informations relatives aux erreurs, si la modification du statut a été déclenchée par une erreur de communication.

7.7.2.4 Service OnFunctionsChanged

Ce service permet à une Application-Cadre de recevoir une notification du DTM stipulant que les informations relatives à son autre fonctionnalité disponible actuelle ont été modifiées. Le Tableau 78 présente les arguments du service.

Tableau 78 – Arguments pour le service OnFunctionsChanged

	Demande	Réponse
fdt:ChannelPath	Ö	–
fdt:SystemTag	M	–

Les DTM doivent utiliser ce service pour informer l'Application-Cadre de mettre à jour ses menus ou ses appels aux fonctions se rapportant à cette fonctionnalité étendue. L'Application-Cadre récupère les fonctions du DTM actuellement disponibles par le service GetFunctions (voir 7.2.5.1).

Si le paramètre ChannelPath est fourni, la fonctionnalité des voies correspondantes a alors été modifiée. Dans ce cas, l'Application-Cadre récupère les fonctions de la voie actuellement disponibles par le service GetChannelFunctions (voir 7.6.3.1).

7.7.3 Services de l'Application-Cadre relatifs à la gestion de la topologie

7.7.3.1 Service GetDtmInfoList

Ce service retourne une liste des informations relatives au DTM. Le Tableau 79 présente les arguments du service.

Tableau 79 – Arguments pour le service GetDtmInfoList

	Demande	Réponse
dtmInfo:DTMInfo list	–	M
dtmInfo.BTMInfo list		M

Ce service retourne une liste des informations relatives au DTM. Ces informations permettent à un DTM d'ajouter un autre DTM à la topologie des FDT en utilisant le service CreateChild (voir 7.7.3.2).

Il incombe à l'Application-Cadre de déterminer quelles informations du DTM sont retournées dans la liste. Par exemple, la liste ne peut contenir que les informations relatives aux DTM pour un protocole de communication même en cas d'installation de DTM pour d'autres protocoles.

7.7.3.2 Service CreateChild

Ce service permet à un DTM d'ajouter un autre DTM à la topologie des FDT. Le Tableau 80 présente les arguments du service.

Tableau 80 – Arguments pour le service CreateChild (DTM)

	Demande	Réponse
fdt:DtmDeviceType	M	–
fdt: ChannelObjectReference	M	–
fdt:SystemTag	–	M

Ce service permet à un DTM d'ajouter un autre DTM identifié par son Type d'Équipement de DTM à la sous-topologie identifiée par le service ChannelObjectReference.

Le DTM est instancié par l'Application-Cadre. Le service retourne le SystemTag du DTM. Si l'opération échoue, aucun SystemTag ne doit être retourné. L'Application-Cadre doit mettre en œuvre le comportement décrit dans le diagramme de séquence en 8.1.2.

Si le service est appelé pour ajouter un BTM, la demande utilise alors le type de bloc du BTM comme cela est défini dans le Tableau 81.

Tableau 81 – Arguments pour le service CreateChild (BTM)

	Demande	Réponse
btm:BtmBlockType	M	–
fdt: ChannelObjectReference	M	–
fdt:SystemTag	–	M

7.7.3.3 Service DeleteChild

Ce service permet à un DTM de retirer un autre DTM de la topologie des FDT. Le Tableau 82 présente les arguments du service.

Tableau 82 – Arguments pour le service DeleteChild

	Demande	Réponse
fdt:SystemTag	M	–
fdt: ChannelObjectReference	M	–
fdt:serviceSucceeded	–	M

Ce service permet à un DTM de retirer un autre DTM spécifié par le SystemTag de la sous-topologie identifiée par le service ChannelObjectReference.

L'Application-Cadre doit appeler le service ValidateRemoveChild (voir 7.6.2.3) au niveau de la Voie de Communication dont le DTM est retiré. S'il s'agit de la dernière référence de la topologie, l'Application-Cadre doit supprimer les données d'instance. L'Application-Cadre doit également appeler le service ClearInstanceData (voir 7.2.4.5) au sujet de ce comportement. L'opération doit également échouer si une sous-topologie existe.

7.7.3.4 Service MoveChild

Ce service permet à un DTM de déplacer un autre DTM dans la topologie des FDT. Le Tableau 83 présente les arguments du service.

Tableau 83 – Arguments pour le service MoveChild

	Demande	Réponse
fdt:SystemTag	M	–
fdt:SystemTag	M	
fdt: ChannelObjectReference	M	–
fdt:serviceSucceeded	–	M

Ce service permet à un DTM de déplacer un autre DTM spécifié par son SystemTag vers une autre sous-topologie identifiée par le SystemTag du DTM de destination et le service ChannelObjectReference.

L'Application-Cadre doit appeler les services ValidateAddChild et ValidateRemoveChild comme cela est défini pour les services CreateChild (voir 7.7.3.2) et DeleteChild (voir 7.7.3.3).

7.7.3.5 Service GetParentNodes

Ce service permet à un DTM de demander une liste des DTM directement supérieurs dans la topologie des FDT. Le Tableau 84 présente les arguments du service.

Tableau 84 – Arguments pour le service GetParentNodes

	Demande	Réponse
fdt:SystemTag	M	–
fdt:SystemTag List	–	M
fdt:SystemTag		M

Le DTM pour lequel la liste doit être créée est identifié par le SystemTag dans la demande. Le service retourne une liste des SystemTag de tous les DTM supérieurs dans la topologie des FDT et un seul SystemTag identifiant le DTM Parent principal.

7.7.3.6 Service GetChildNodes

Ce service permet à un DTM de demander une liste des DTM directement inférieurs dans la topologie des FDT. Le Tableau 85 présente les arguments du service.

Tableau 85 – Arguments pour le service GetChildNodes

	Demande	Réponse
fdt:SystemTag	M	–
fdt: ChannelObjectReference	M	–
fdt:SystemTag List	–	M

Le DTM et la Voie de Communication pour lesquels la liste doit être créée sont identifiés par le SystemTag et un service ChannelObjectReference. Le service retourne une liste des SystemTag de tous les DTM inférieurs dans la topologie des FDT.

7.7.3.7 Service GetDtm

Ce service permet à un DTM de demander une référence à un autre DTM identifié par son SystemTag. Le Tableau 86 présente les arguments du service.

Tableau 86 – Arguments pour le service GetDtm

	Demande	Réponse
fdt:SystemTag	M	–
fdt: DtmObjectReference	–	M

Ce service permet à un DTM de demander une référence à un autre DTM identifié par son SystemTag. Les appels supplémentaires au sein d'une instance FrameApplication doivent retourner l'ObjectReference du DTM identique.

Le DTM qui a demandé les références doit appeler le service ReleaseDtm (voir 7.7.3.8) afin de libérer la référence. L'Application-Cadre doit mettre en œuvre un type de comptage des références exigé pour gérer plusieurs références à l'instance d'un même DTM.

7.7.3.8 Service ReleaseDtm

Ce service doit être utilisé pour libérer la référence au DTM associé qui a été demandée par le service GetDtm (voir 7.7.3.7). Le Tableau 87 présente les arguments du service.

Tableau 87 – Arguments pour le service ReleaseDtm

	Demande	Réponse
fdt:SystemTag	M	–
fdt:serviceSucceeded	–	M

Ce service doit être utilisé pour libérer la référence au DTM associé, identifié par son SystemTag, dont le service GetDtm a fait la demande (voir 7.7.3.7).

7.7.4 Services de l'Application-Cadre relatifs à la redondance

7.7.4.1 Service OnAddedRedundantChild

Un DTM Parent doit envoyer ces informations à l'Application-Cadre si un autre DTM qui gère un équipement redondant est ajouté à la topologie. L'Application-Cadre est alors capable d'afficher l'instance au niveau d'une autre Voie de Communication redondante. Le Tableau 88 présente les arguments du service.

Tableau 88 – Arguments pour le service OnAddedRedundantChild

	Demande	Réponse
fdt:SystemTag	M	–
fdt:ChannelObjectReference	M	–
fdt:serviceSucceeded	–	M

Le DTM Parent a ajouté le DTM identifié par son SystemTag, qui gère un esclave redondant, à la Voie de Communication identifiée par le service ChannelObjectReference. L'Application-Cadre ne doit pas appeler le service ValidateAddChild (voir 7.6.2.1) ou ChildAdded (voir 7.6.2.2) sur cette voie.

7.7.4.2 Service OnRemovedRedundantChild

Un DTM Parent doit envoyer ces informations à l'Application-Cadre si un autre DTM qui gère un équipement redondant est retiré de la topologie. L'Application-Cadre est capable de cacher l'instance au niveau de l'autre Voie de Communication redondante. Le Tableau 89 présente les arguments du service.

Tableau 89 – Arguments pour le service OnRemovedRedundantChild

	Demande	Réponse
fdt:SystemTag	M	–
fdt:ChannelObjectReference	M	–
fdt:serviceSucceeded	–	M

Le DTM Parent retire le DTM identifié par son SystemTag, qui gère un esclave redondant, de la Voie de Communication identifiée par le service ChannelObjectReference. L'Application-Cadre ne doit pas appeler le service ValidateAddChild (voir 7.6.2.3) ou ChildAdded (voir 7.6.2.4) sur cette voie.

7.7.5 Services de l'Application-Cadre relatifs au stockage des données du DTM

7.7.5.1 Service SaveInstanceData

Ce service permet à un DTM de sauvegarder ses données d'instance dans la mémoire du Projet géré par l'Application-Cadre (voir le 4.9). Le Tableau 90 présente les arguments du service.

Tableau 90 – Arguments pour le service SaveInstanceData

	Demande	Réponse
fdt:StorageObject	M	–
fdt:serviceSucceeded	–	M

Il incombe à l'Application-Cadre de déterminer si les données transmises dans les demandes sont immédiatement stockées ou non dans la mémoire du Projet ou mises en mémoire cache jusqu'à l'occurrence d'un autre événement (par exemple, une demande explicite de l'utilisateur).

Le DTM qui appelle ce service doit maintenir le verrouillage des données d'instance (voir service LockInstanceData, 7.7.6.1). À défaut, l'appel au service échoue (retourne le message fdtServiceSucceeded = FALSE).

7.7.5.2 Service LoadInstanceData

Ce service permet à un DTM de charger des données d'instance à partir de la mémoire du Projet géré par l'Application-Cadre (voir le 4.9). Le Tableau 91 présente les arguments du service.

Tableau 91 – Arguments pour le service LoadInstanceData

	Demande	Réponse
fdt:StorageObject	–	M

Le DTM qui appelle ce service récupère exactement les mêmes données que celles stockées par le service SaveInstanceData (voir 7.7.5.1).

7.7.5.3 Service GetPrivateDtmStorageInformation

Ce service permet à un DTM de demander à l'Application-Cadre des informations relatives au stockage privé de ses données (voir le 4.9). Le Tableau 92 présente les arguments du service.

Tableau 92 – Arguments pour le service GetPrivateDtmStorageInformation

	Demande	Réponse
InstanceRelatedStorageInfo	–	M
ProjectRelatedStorageInfo	–	M

Le paramètre de réponse InstanceRelatedStorageInfo comporte les informations qui permettent à un DTM d'accéder à la mémoire appartenant à l'instance appelante. Le paramètre de réponse ProjectRelatedStorageInfo comporte les informations permettant d'accéder à la mémoire partagée par toutes les instances de ce type de DTM dans un Projet. Le mécanisme de stockage privé du DTM dépend de la technologie de mise en œuvre et est ainsi défini dans le document relatif au profil d'intégration des modèles d'ojets de la série IEC 62453 qui définit le mapping avec des technologies particulières.

Si une Application-Cadre n'est pas capable de fournir un accès aux mémoires de données privées du DTM, elle ne retourne alors aucune information. Un DTM doit être capable de fonctionner sans effet secondaire dans ce cas. Il doit garantir l'absence d'incidence sur les données d'instance gérées par les services SaveInstanceData (voir 7.7.5.1) et LoadInstanceData (voir 7.7.5.2).

Un DTM doit nettoyer la mémoire relative aux instances si le service ClearInstanceData (voir 7.2.4.5) est appelé. Si le DTM comporte des références entre les mémoires du Projet et celles relatives aux instances, il doit alors également nettoyer ces mémoires dans la mémoire du Projet.