



BSI Standards Publication

Field device tool (FDT) interface specification

Part 2: Concepts and detailed description

National foreword

This British Standard is the UK implementation of EN 62453-2:2022. It is identical to IEC 62453-2:2022. It supersedes BS EN 62453-2:2017, which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee GEL/65/3, Industrial communications: process measurement and control, including fieldbus.

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

Contractual and legal considerations

This publication has been prepared in good faith, however no representation, warranty, assurance or undertaking (express or implied) is or will be made, and no responsibility or liability is or will be accepted by BSI in relation to the adequacy, accuracy, completeness or reasonableness of this publication. All and any such responsibility and liability is expressly disclaimed to the full extent permitted by the law.

This publication is provided as is, and is to be used at the recipient's own risk.

The recipient is advised to consider seeking professional guidance with respect to its use of this publication.

This publication is not intended to constitute a contract. Users are responsible for its correct application.

© The British Standards Institution 2022
Published by BSI Standards Limited 2022

ISBN 978 0 539 15387 3

ICS 25.040.40; 35.100.05; 35.110

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

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 October 2022.

Amendments/corrigenda issued since publication

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

English Version

Field device tool (FDT) interface specification - Part 2: Concepts
and detailed description
(IEC 62453-2:2022)

Spécification des interfaces des outils des dispositifs de
terrain (FDT) - Partie 2: Concepts et description détaillée
(IEC 62453-2:2022)

Field Device Tool (FDT)-Schnittstellenspezifikation - Teil 2:
Konzept und grundlegende Beschreibung
(IEC 62453-2:2022)

This European Standard was approved by CENELEC on 2022-10-05. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

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

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

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



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

CEN-CENELEC Management Centre: Rue de la Science 23, B-1040 Brussels

European foreword

The text of document 65E/906/FDIS, future edition 3 of IEC 62453-2, prepared by SC 65E "Devices and integration in enterprise systems" of IEC/TC 65 "Industrial-process measurement, control and automation" was submitted to the IEC-CENELEC parallel vote and approved by CENELEC as EN IEC 62453-2:2022.

The following dates are fixed:

- latest date by which the document has to be implemented at national level by publication of an identical national standard or by endorsement (dop) 2023-07-05
- latest date by which the national standards conflicting with the document have to be withdrawn (dow) 2025-10-05

This document supersedes EN 62453-2:2017 and all of its amendments and corrigenda (if any).

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

Any feedback and questions on this document should be directed to the users' national committee. A complete listing of these bodies can be found on the CENELEC website.

Endorsement notice

The text of the International Standard IEC 62453-2:2022 was approved by CENELEC as a European Standard without any modification.

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

IEC/TR 62453-41 NOTE Harmonized as CLC/TR IEC 62453-41

IEC/TR 62453-42 NOTE Harmonized as CLC/TR IEC 62453-42

Annex ZA (normative)

Normative references to international publications with their corresponding European publications

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.

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

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

<u>Publication</u>	<u>Year</u>	<u>Title</u>	<u>EN/HD</u>	<u>Year</u>
IEC 61131	series	Programmable controllers	EN 61131	series
IEC/TR 62390	2005	Common automation device - Profile guideline	-	-
IEC 62453-1	2016	Field device tool (FDT) interface specification - Part 1: Overview and guidance	EN 62453-1	2017
IEC 62453-3xy	series	Field device tool (FDT) interface specification - Part 3xy: Communication profile integration	EN 62453-3xy	series

CONTENTS

FOREWORD	10
INTRODUCTION	12
1 Scope	13
2 Normative references	13
3 Terms, definitions, symbols, abbreviated terms and conventions	13
3.1 Terms and definitions	13
3.2 Symbols and abbreviated terms	14
3.3 Conventions	14
3.3.1 Use of UML	14
3.3.2 State availability statement	14
3.3.3 Data type names and references to data types	14
4 Fundamentals	14
4.1 General	14
4.2 Abstract FDT model	15
4.2.1 FDT model overview	15
4.2.2 Frame Application (FA)	18
4.2.3 Device Type Manager (DTM)	19
4.2.4 Channel object	26
4.3 Modularity	28
4.4 Bus categories	28
4.5 Identification	29
4.5.1 DTM instance identification	29
4.6 System and FDT topology	30
4.7 FDT Communication	31
4.7.1 General	31
4.7.2 Handling of communication requests	32
4.7.3 Handling of communication errors	32
4.7.4 Handling of loss of connection	32
4.7.5 Point-to-point communication	32
4.7.6 Nested communication	33
4.8 DTM, DTM Device Type and hardware identification information	35
4.8.1 DTM and DTM Device Type	35
4.8.2 Supported hardware identification	36
4.8.3 Connected hardware identification	36
4.9 DTM data persistence and synchronization	37
4.10 DTM device parameter access	38
4.11 DTM state machine	38
4.11.1 DTM states	38
4.11.2 'Communication allowed' substates	39
4.12 Basic operation phases	40
4.12.1 Roles and access rights	40
4.12.2 Operation phases	40
4.13 FDT version interoperability	41
4.13.1 Version interoperability overview	41
4.13.2 DTM and device versions	42
4.13.3 Persistence	42

4.13.4	Nested communication	43
5	FDT session model and use cases	43
5.1	Session model overview.....	43
5.2	Actors	44
5.3	Use cases	46
5.3.1	Use case overview.....	46
5.3.2	Observation	46
5.3.3	Operation	46
5.3.4	Maintenance	50
5.3.5	Planning	53
5.3.6	OEM service.....	56
5.3.7	Administration.....	56
6	General concepts	57
6.1	Address management	57
6.2	Scanning and DTM assignment.....	58
6.2.1	Scanning overview.....	58
6.2.2	Scanning	58
6.2.3	DTM assignment.....	59
6.2.4	Manufacturer-specific device identification.....	59
6.2.5	Scan for communication hardware	60
6.3	Configuration of Fieldbus Master or Communication Scheduler.....	60
6.4	PLC tool support.....	61
6.4.1	General	61
6.4.2	Process image modifications while PLC is running.....	62
6.5	Slave redundancy	63
6.5.1	Redundancy overview.....	63
6.5.2	Redundancy support in Frame Application	64
6.5.3	Parent component for redundant fieldbus.....	64
6.5.4	Redundancy support in Device DTM	65
6.5.5	Scan and redundant slaves.....	65
7	FDT service specification.....	65
7.1	Service specification overview	65
7.2	DTM services.....	66
7.2.1	General services.....	66
7.2.2	DTM services related to installation	68
7.2.3	DTM services related to DTM/device information	69
7.2.4	DTM services related to the DTM state machine	71
7.2.5	DTM services related to functions	74
7.2.6	DTM services related to Channel objects – service GetChannels	77
7.2.7	DTM services related to documentation – service GetDocumentation	77
7.2.8	DTM services to access the instance data	77
7.2.9	DTM services to evaluate the instance data.....	79
7.2.10	DTM services to access the device data	80
7.2.11	DTM services related to network management information	81
7.2.12	DTM services related to online operation	82
7.2.13	DTM services related to data synchronization	84
7.2.14	DTM services related to import and export.....	86
7.3	Presentation object services	86
7.4	Channel object service.....	87

7.4.1	Channel object service overview	87
7.4.2	Service ReadChannelInformation	87
7.4.3	Service WriteChannelInformation	87
7.5	Process Channel object services – services for I/O related information	87
7.5.1	Service ReadChannelData	87
7.5.2	Service WriteChannelData	88
7.6	Communication Channel object services	88
7.6.1	Services related to communication	88
7.6.2	Services related to sub-topology management	92
7.6.3	Services related to GUI and functions	94
7.6.4	Service Scan	95
7.7	Frame Application services	96
7.7.1	General state availability	96
7.7.2	FA services related to general events	96
7.7.3	FA services related to topology management	97
7.7.4	FA services related to redundancy	100
7.7.5	FA services related to storage of DTM data	101
7.7.6	FA services related to DTM data synchronization	102
7.7.7	FA service related to process image validation – service ValidateProcessImage	103
7.7.8	FA services related to presentation	104
7.7.9	FA Services related to audit trail – service RecordAuditTrailEvent	105
8	FDT dynamic behavior	105
8.1	Generate FDT topology	105
8.1.1	FDT topology generation triggered by the Frame Application	105
8.1.2	FDT topology generation triggered by the DTM	106
8.2	Address setting	107
8.2.1	Address setting overview	107
8.2.2	Set or modify device address – with user interface	107
8.2.3	Set or modify device address – without user interface	107
8.2.4	Display or modify all child device addresses with user interface	108
8.3	Communication	109
8.3.1	Communication overview	109
8.3.2	Point-to-point communication	109
8.3.3	Nested communication	109
8.3.4	Device-initiated data transfer	110
8.4	Scanning and DTM assignment	111
8.5	Multi-user scenarios	112
8.5.1	General	112
8.5.2	Synchronized and non-synchronized locking mechanism for DTMs	114
8.5.3	Additional rules	116
8.6	Notification of changes	116
8.7	DTM instance data state machines	116
8.7.1	Instance data set overview	116
8.7.2	Modifications state machine	117
8.7.3	Persistence state machine	118
8.7.4	Modification in device	118
8.7.5	Storage life cycle	119
8.8	Parent component handling redundant slave	

8.9	DTM upgrade	121
8.9.1	General rules	121
8.9.2	Saving data from a DTM to be upgraded	122
8.9.3	Loading data in the replacement DTM	122
Annex A	(normative) FDT data types definition	124
A.1	General	124
A.2	Basic data types	125
A.3	General data types	125
A.4	User information data types	141
A.5	DTM information data type	142
A.6	BTM data types	142
A.7	Device and Scan identification data types	144
A.8	Function data types	148
A.9	AuditTrail data types	151
A.10	Documentation data types	152
A.11	DeviceList data type	153
A.12	Network management data types	155
A.13	Instance data types	156
A.14	DeviceStatus data types	161
A.15	OnlineCompare data types	161
A.16	UserInterface data types	162
A.17	Fieldbus-specific data types	163
Bibliography		165
Figure 1	– Part 2 of the IEC 62453 series	12
Figure 2	– Abstract FDT model	15
Figure 3	– Frame Application with integrated Communication Channel	19
Figure 4	– Device Type Manager (DTM)	19
Figure 5	– Communication DTM	20
Figure 6	– Device DTM	21
Figure 7	– Gateway DTM	21
Figure 8	– Composite Device DTM	22
Figure 9	– Module DTM	23
Figure 10	– Block Type Manager (BTM)	24
Figure 11	– Presentation object	25
Figure 12	– Channel object	26
Figure 13	– Communication Channel	27
Figure 14	– Combined Process/Communication Channel	28
Figure 15	– Identification of connected devices	29
Figure 16	– FDT topology for a simple system topology	30
Figure 17	– FDT topology for a complex system topology	31
Figure 18	– Point-to-point communication	33
Figure 19	– Nested communication	34
Figure 20	– DTM, DTM Device Type and device identification information	35
Figure 21	– Connected hardware identification	36
Figure 22	– FDT storage and synchronization mechanisms	37

Figure 23 – DTM state machine	38
Figure 24 – Substates of communication allowed	39
Figure 25 – Main use case diagram	44
Figure 26 – Observation use cases	46
Figure 27 – Operation use cases	47
Figure 28 – Maintenance use cases	50
Figure 29 – Planning use cases	54
Figure 30 – OEM Service	56
Figure 31 – Administrator use cases	56
Figure 32 – Address setting via DTM Presentation object	58
Figure 33 – Fieldbus scanning	59
Figure 34 – Fieldbus master configuration tool as part of a DTM	61
Figure 35 – Process Image	62
Figure 36 – Transfer of layout information using ProcessImage services	62
Figure 37 – Redundancy scenarios	63
Figure 38 – FDT topology generation triggered by the Frame Applications	106
Figure 39 – FDT topology generation triggered by a DTM	106
Figure 40 – Set or modify device address – with user interface	107
Figure 41 – Set or modify device address – without user interface	108
Figure 42 – Set or modify all device addresses – with user interface	108
Figure 43 – Point-to-point communication	109
Figure 44 – Nested communication	110
Figure 45 – Device-initiated data transfer	111
Figure 46 – Scanning and DTM assignment	112
Figure 47 – Multi-user system	113
Figure 48 – General synchronized locking mechanism	114
Figure 49 – General non-synchronized locking mechanism	115
Figure 50 – Parameterization in the case of synchronized locking mechanism	115
Figure 51 – Modifications state machine of instance data	117
Figure 52 – Persistence state machine of instance data	118
Figure 53 – Management of redundant topology	121
Figure 54 – Associating data to a dataSetId	122
Figure 55 – Loading data for a supported dataSetId	123
Table 1 – Description of FDT objects	16
Table 2 – Description of associations between FDT objects	17
Table 3 – Transitions of DTM states	39
Table 4 – Transitions of DTM 'communication allowed' substates	40
Table 5 – Operation phases	41
Table 6 – Actors	45
Table 7 – Operation use cases	48
Table 8 – Maintenance use cases	51
Table 9 – Planning use cases	54

Table 10 – Administrator use cases	57
Table 11 – Arguments for service PrivateDialogEnabled	66
Table 12 – Arguments for service SetLanguage	67
Table 13 – Arguments for service SetSystemGuiLabel	68
Table 14 – Arguments for service GetTypeInformation (for DTM)	69
Table 15 – Arguments for service GetTypeInformation (for BTM)	69
Table 16 – Arguments for service GetIdentificationInformation (for DTM)	69
Table 17 – Arguments for service GetIdentificationInformation (for BTM)	70
Table 18 – Arguments for service Hardware information (for DTM)	70
Table 19 – Arguments for service GetActiveTypeInfo	70
Table 20 – Arguments for service GetActiveTypeInfo (for BTM)	71
Table 21 – Arguments for service Initialize (for DTM)	71
Table 22 – Arguments for service Initialize (for BTM)	72
Table 23 – Arguments for service SetLinkedCommunicationChannel	72
Table 24 – Arguments for service EnableCommunication	72
Table 25 – Arguments for service ReleaseLinkedCommunicationChannel	73
Table 26 – Arguments for service ClearInstanceData	73
Table 27 – Arguments for service Terminate	74
Table 28 – Arguments for service GetFunctions	74
Table 29 – Arguments for service InvokeFunctions	75
Table 30 – Arguments for service GetGuiInformation	75
Table 31 – Arguments for service OpenPresentation	76
Table 32 – Arguments for service ClosePresentation	76
Table 33 – Arguments for service GetChannels	77
Table 34 – Arguments for service GetDocumentation	77
Table 35 – Arguments for service InstanceDataInformation	78
Table 36 – Arguments for service InstanceDataRead	78
Table 37 – Arguments for service InstanceDataWrite	79
Table 38 – Arguments for service Verify	79
Table 39 – Arguments for service CompareDataValueSets	79
Table 40 – Arguments for service DeviceDataInformation	80
Table 41 – Arguments for service DeviceDataRead	80
Table 42 – Arguments for service DeviceDataWrite	81
Table 43 – Arguments for service NetworkManagementInfoRead	81
Table 44 – Arguments for service NetworkManagementInfoWrite	82
Table 45 – Arguments for service DeviceStatus (for DTM)	82
Table 46 – Arguments for service CompareInstanceDataWithDeviceData (for DTM)	83
Table 47 – Arguments for service WriteDataToDevice (for DTM)	83
Table 48 – Arguments for service ReadDataFromDevice (for DTM)	84
Table 49 – Arguments for service OnLockInstanceData	84
Table 50 – Arguments for service OnUnlockInstanceData	85
Table 51 – Arguments for service OnInstanceDataChanged	85
Table 52 – Arguments for service OnInstanceChildDataChanged	85

Table 53 – Arguments for service Export	86
Table 54 – Arguments for service Import.....	86
Table 55 – Arguments for service ReadChannelInformation	87
Table 56 – Arguments for service WriteChannelInformation	87
Table 57 – Arguments for service ReadChannelData	87
Table 58 – Arguments for service WriteChannelData	88
Table 59 – Arguments for service GetSupportedProtocols.....	88
Table 60 – Arguments for service Connect.....	89
Table 61 – Arguments for service Disconnect	90
Table 62 – Arguments for service AbortRequest	90
Table 63 – Arguments for service AbortIndication	90
Table 64 – Arguments for service Transaction	91
Table 65 – Arguments for service SequenceDefine	92
Table 66 – Arguments for service SequenceStart.....	92
Table 67 – Arguments for service ValidateAddChild	92
Table 68 – Arguments for service ChildAdded.....	93
Table 69 – Arguments for service ValidateRemoveChild	93
Table 70 – Arguments for service ChildRemoved	94
Table 71 – Arguments for service SetChildrenAddresses	94
Table 72 – Arguments for service GetChannelFunctions	95
Table 73 – Arguments for service GetGuiInformation	95
Table 74 – Arguments for service Scan.....	95
Table 75 – Arguments for service OnErrorMessage	96
Table 76 – Arguments for service OnProgress	96
Table 77 – Arguments for service OnOnlineStatusChanged	97
Table 78 – Arguments for service OnFunctionsChanged	97
Table 79 – Arguments for service GetDtmInfoList	97
Table 80 – Arguments for service CreateChild (DTM)	98
Table 81 – Arguments for service CreateChild (BTM).....	98
Table 82 – Arguments for service DeleteChild.....	98
Table 83 – Arguments for service MoveChild	99
Table 84 – Arguments for service GetParentNodes	99
Table 85 – Arguments for service GetChildNodes	99
Table 86 – Arguments for service GetDtm.....	100
Table 87 – Arguments for service ReleaseDtm.....	100
Table 88 – Arguments for service OnAddedRedundantChild	100
Table 89 – Arguments for service OnRemovedRedundantChild.....	101
Table 90 – Arguments for service SaveInstanceData	101
Table 91 – Arguments for service LoadInstanceData	101
Table 92 – Arguments for service GetPrivateDtmStorageInformation	102
Table 93 – Arguments for service LockInstanceData	102
Table 94 – Arguments for service UnlockInstanceData.....	103
Table 95 – Arguments for service OnInstanceDataChanged.....	103

Table 96 – Arguments for service ValidateProcessImage	103
Table 97 – Arguments for service OpenPresentationRequest	104
Table 98 – Arguments for service ClosePresentationRequest	104
Table 99 – Arguments for service UserDialog	105
Table 100 – Arguments for service RecordAuditTrailEvent	105
Table 101 – Modifications state machine of instance data	117
Table 102 – Persistence state machine of instance data	118
Table 103 – Example life cycle of a DTM	119
Table A.1 – Basic data types	125
Table A.2 – Simple general data types	125
Table A.3 – Definition of classificationId enumeration values	132
Table A.4 – General structured data types	134
Table A.5 – Simple user information data types	141
Table A.6 – Structured user information data type	142
Table A.7 – Structured DTM information data type	142
Table A.8 – Simple BTM data types	143
Table A.9 – Structured BTM data types	144
Table A.10 – Simple device identification data types	145
Table A.11 – Structured device identification data types	146
Table A.12 – Simple function data types	149
Table A.13 – Structured function data types	150
Table A.14 – Simple auditTrail data types	151
Table A.15 – Structured auditTrail data types	152
Table A.16 – Simple documentation data types	152
Table A.17 – Structured documentation data types	153
Table A.18 – Simple deviceList data type	154
Table A.19 – Structured deviceList data type	154
Table A.20 – Simple network management data types	155
Table A.21 – Structured network management data types	156
Table A.22 – Simple instance data types	157
Table A.23 – Structured instance data types	159
Table A.24 – Simple device status data types	161
Table A.25 – Structured device status data types	161
Table A.26 – Simple online compare data types	161
Table A.27 – Structured online compare data types	162
Table A.28 – Simple user interface data types	162
Table A.29 – Structured user interface data types	163
Table A.30 – Fieldbus data types	164

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 FDT¹ (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.

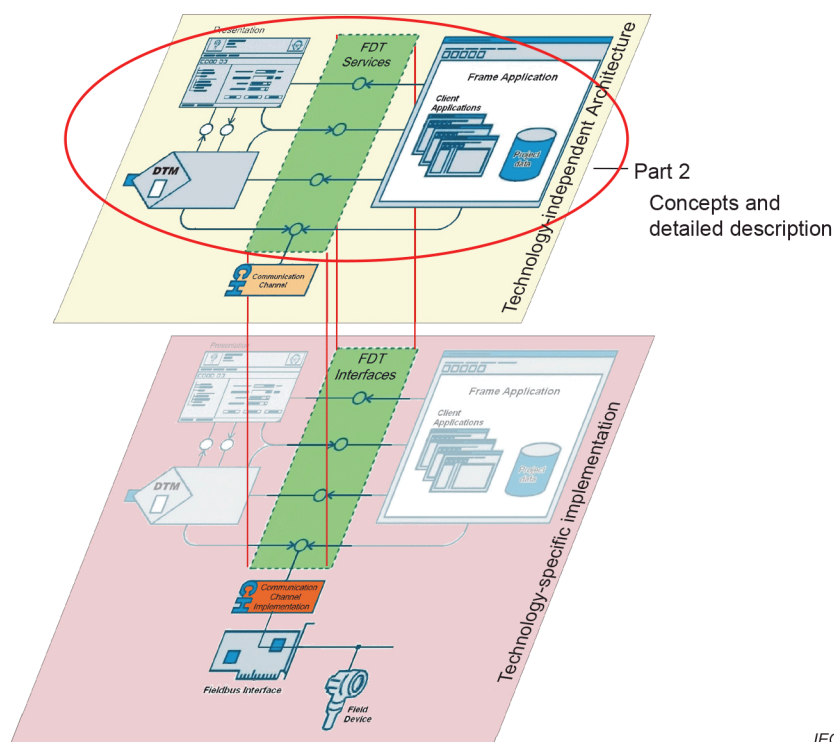


Figure 1 – Part 2 of the IEC 62453 series

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

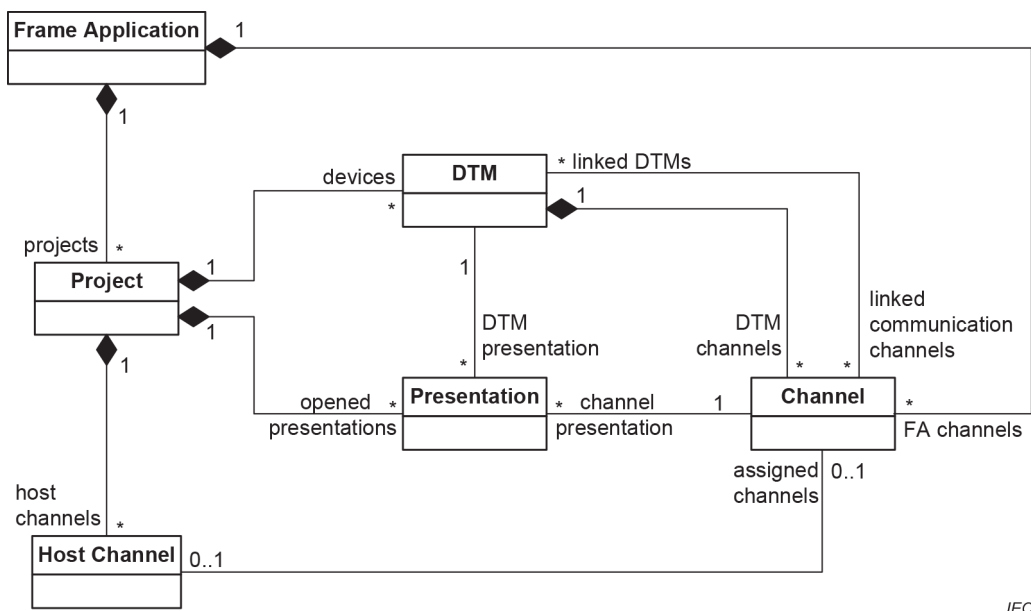
4.1 General

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.



IEC

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. <u>Use cases:</u> – channel assignment <u>Services:</u> 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. <u>Use cases:</u> – all channel use cases with DTM-specific GUI <u>Services:</u> Informational services (see also 7.3) – channel service: GetChannelFunctions – 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. <u>Use cases:</u> – system generation – system planning <u>Services:</u> 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. <u>Services:</u> 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. <u>Use cases:</u> – All DTM use cases with DTM-specific GUI <u>Services:</u> – 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. <u>Services:</u> – 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. <u>Services:</u> – 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 to perform low-level actions such as monitoring or filtering. Frame Applications without built-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.

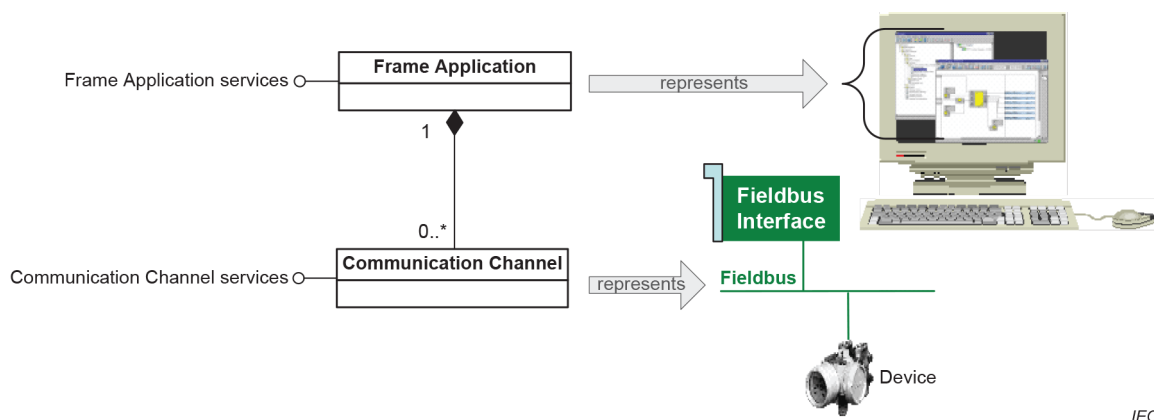


Figure 3 – Frame Application with integrated Communication Channel

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.

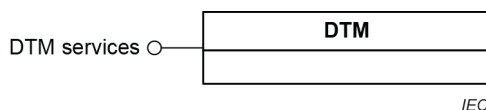


Figure 4 – Device Type Manager (DTM)

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).

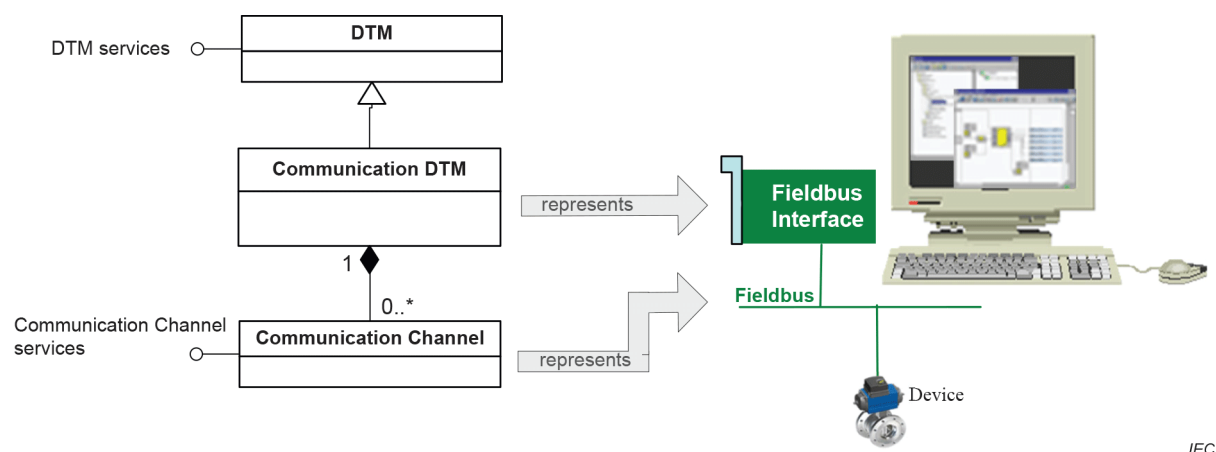


Figure 5 – Communication DTM

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).

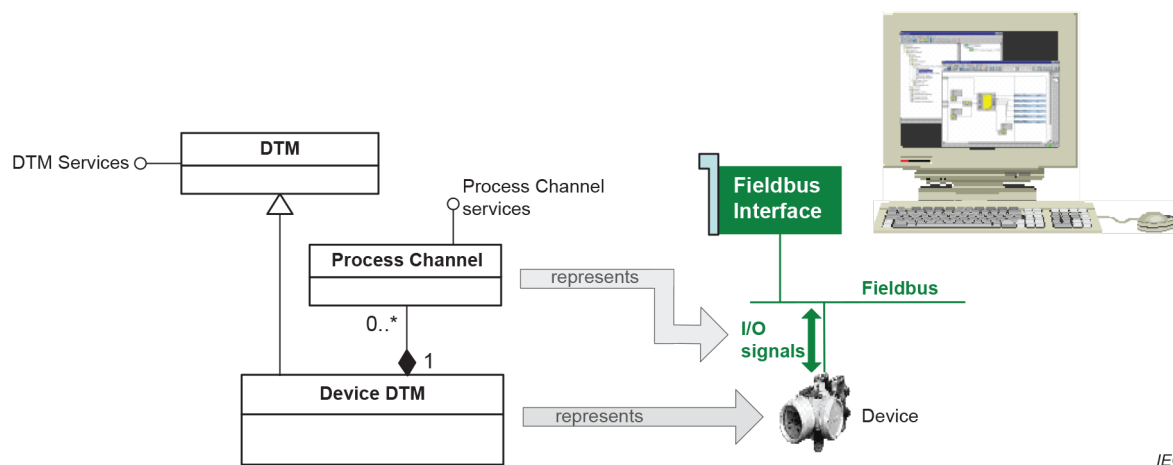


Figure 6 – Device DTM

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).

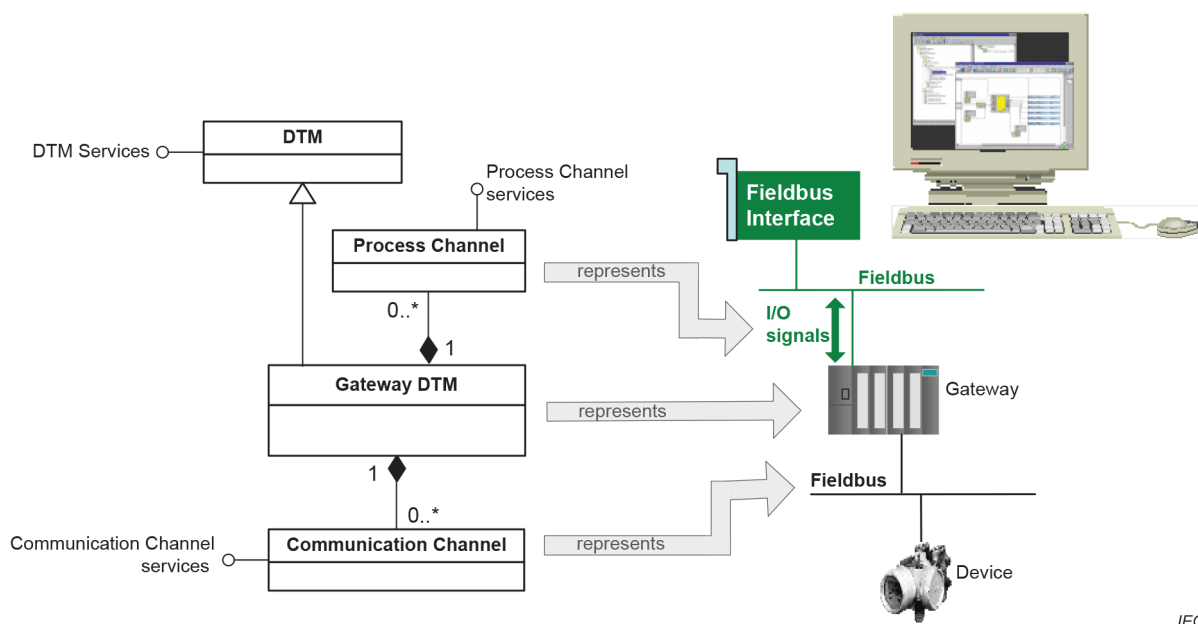


Figure 7 – Gateway DTM

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.

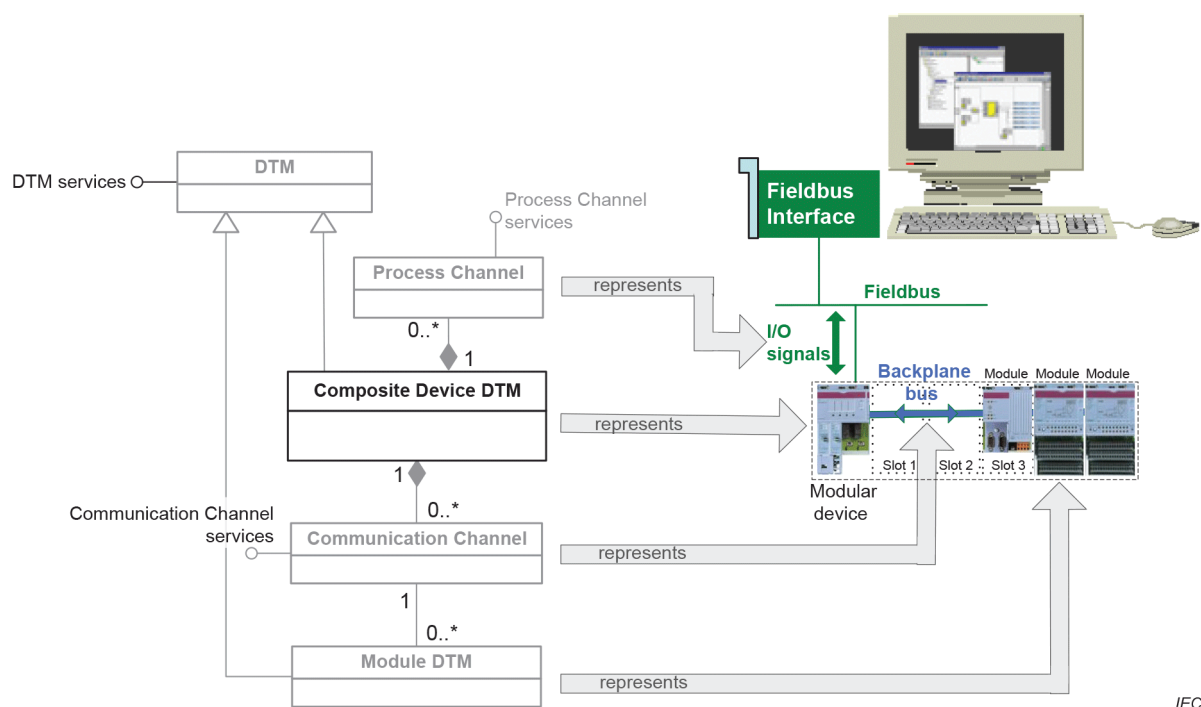


Figure 8 – Composite Device DTM

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.

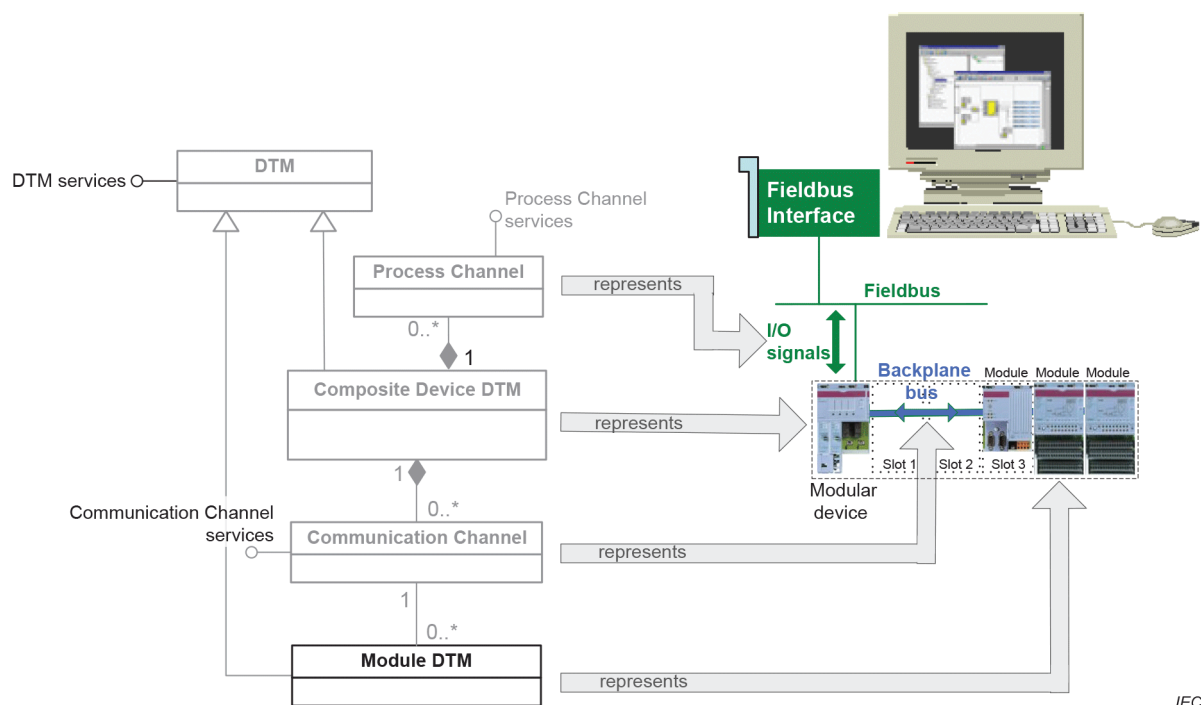


Figure 9 – Module DTM

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).

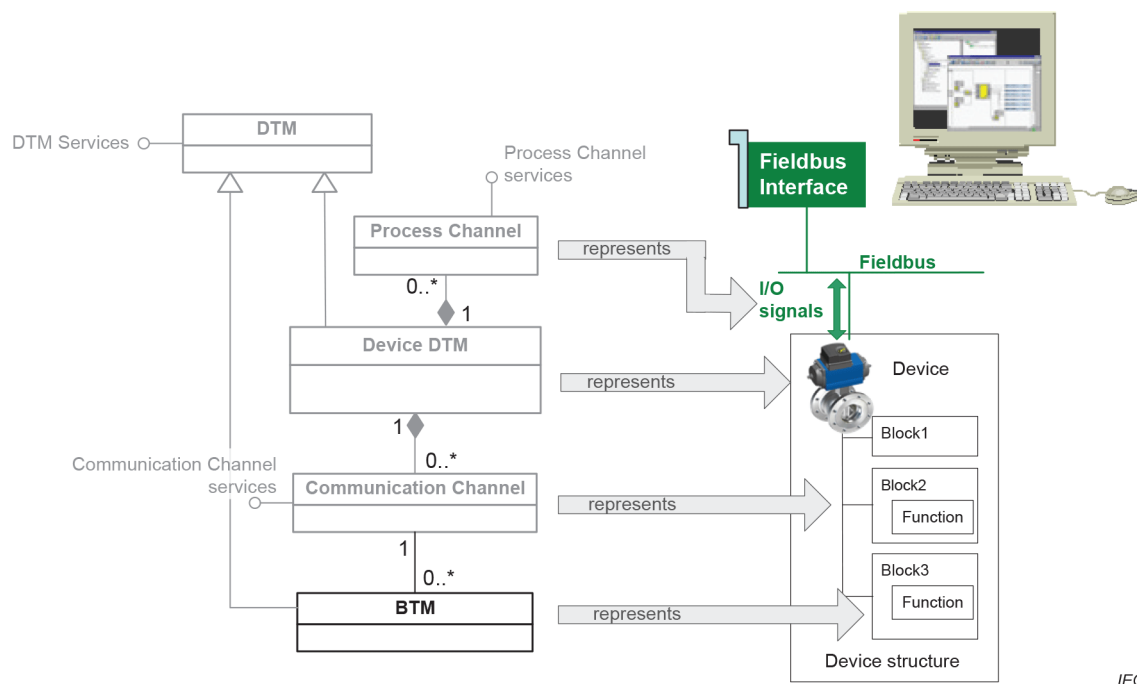


Figure 10 – Block Type Manager (BTM)

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 separately 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).

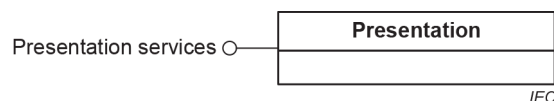


Figure 11 – Presentation object

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.

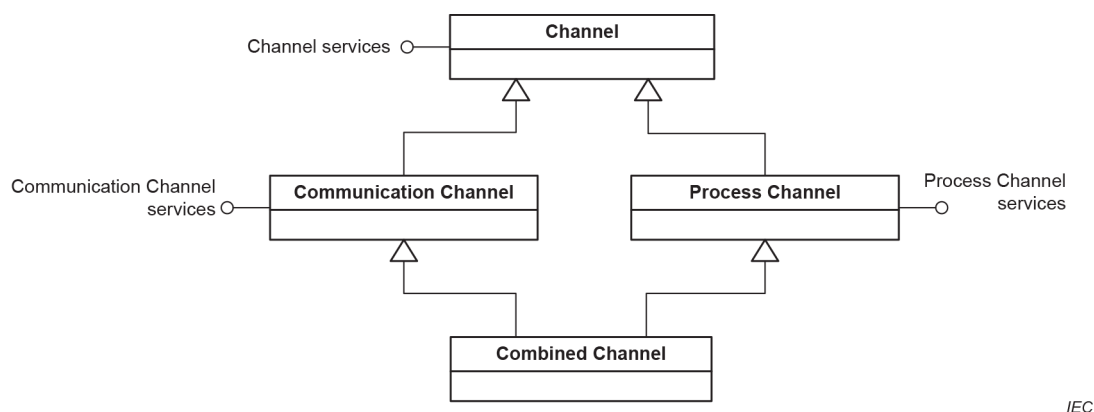


Figure 12 – Channel object

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).

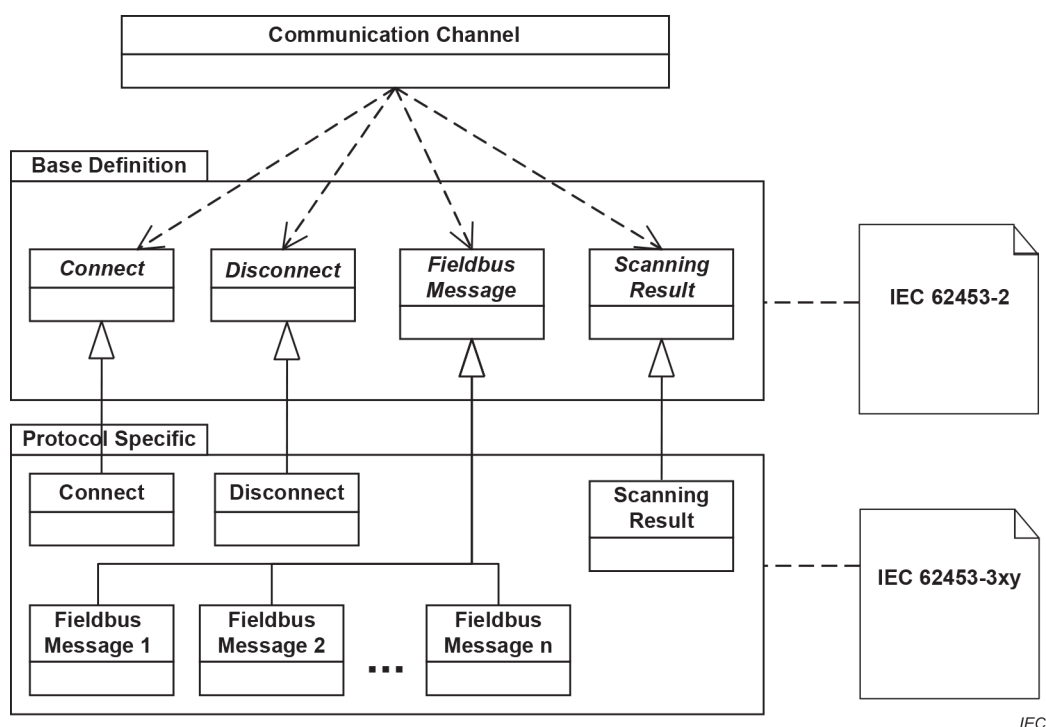


Figure 13 – Communication Channel

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.

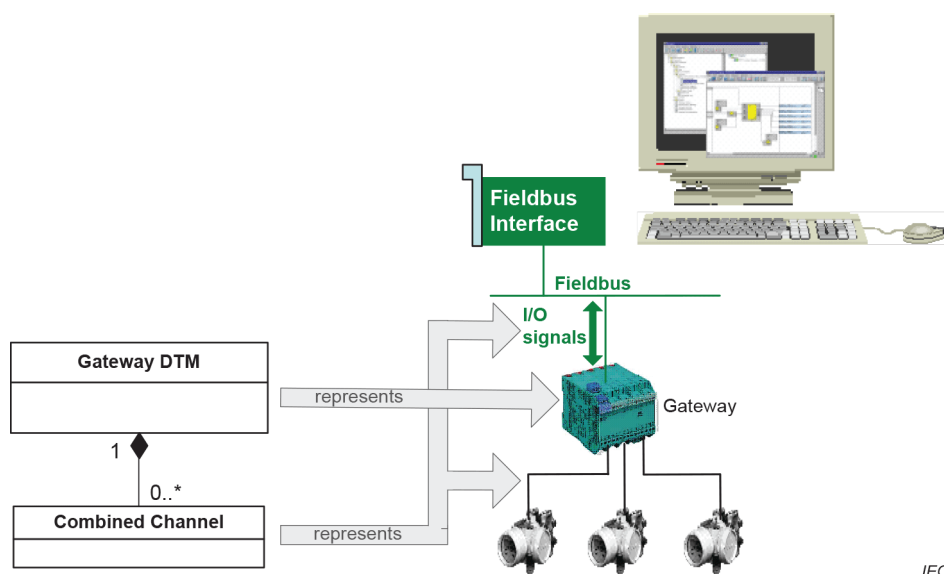


Figure 14 – Combined Process/Communication Channel

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).

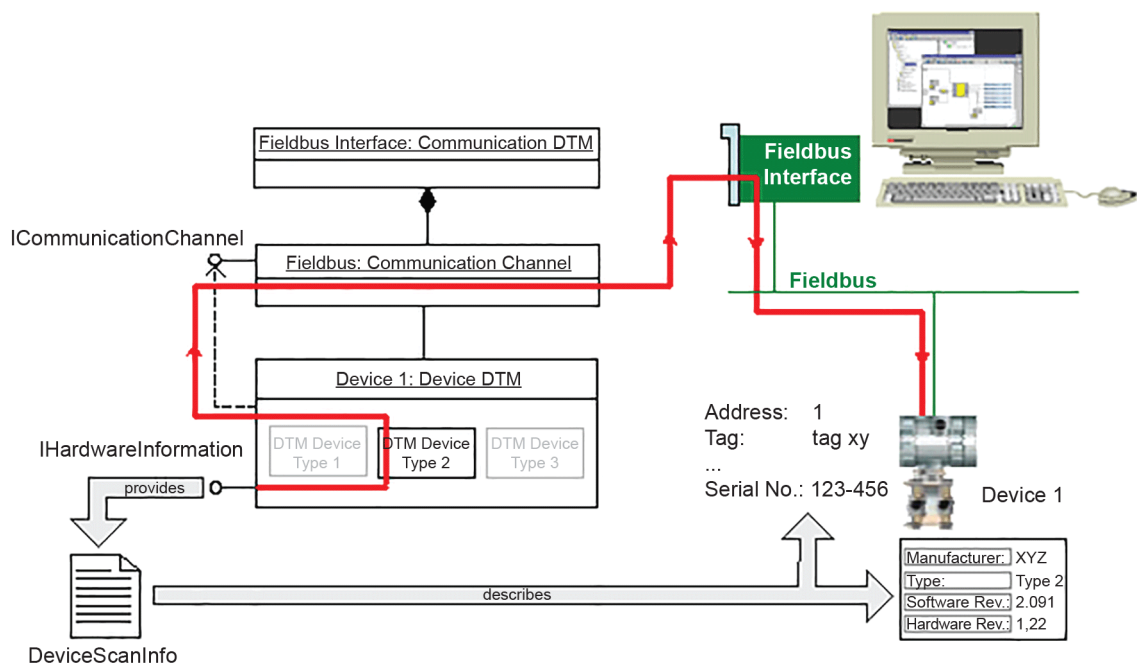
4.5.1.2 System GUI label

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).



IEC

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.

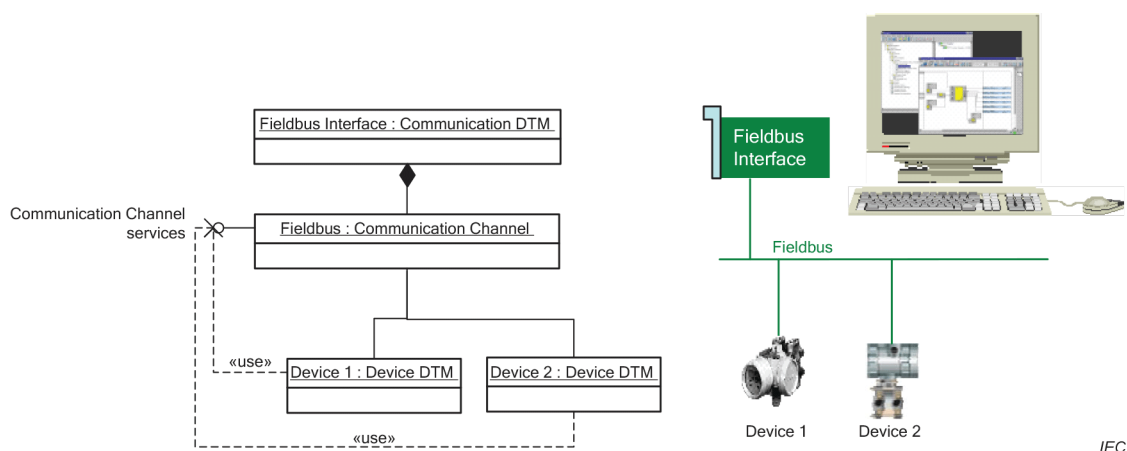


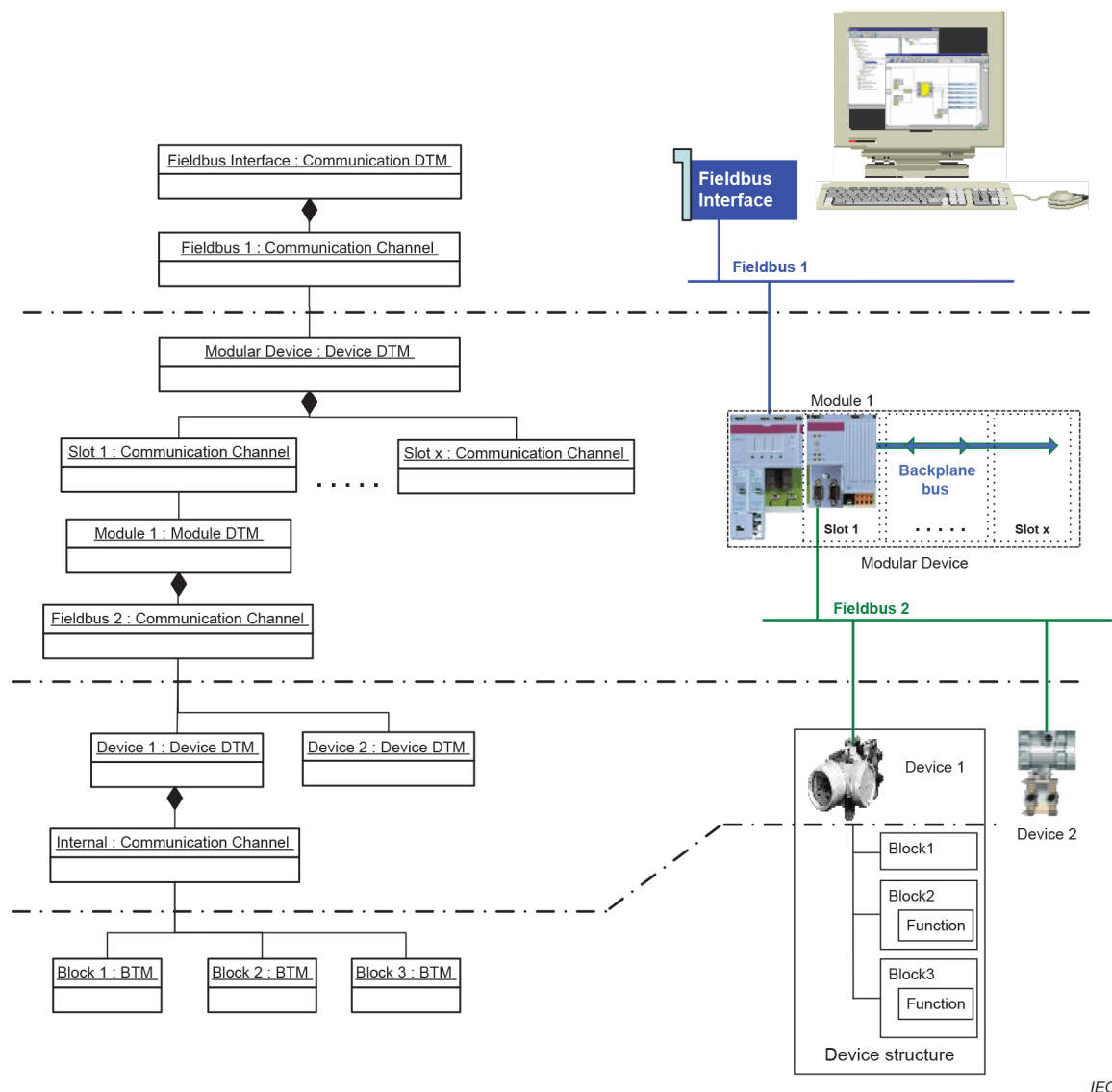
Figure 16 – FDT topology for a simple system topology

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.



IEC

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.

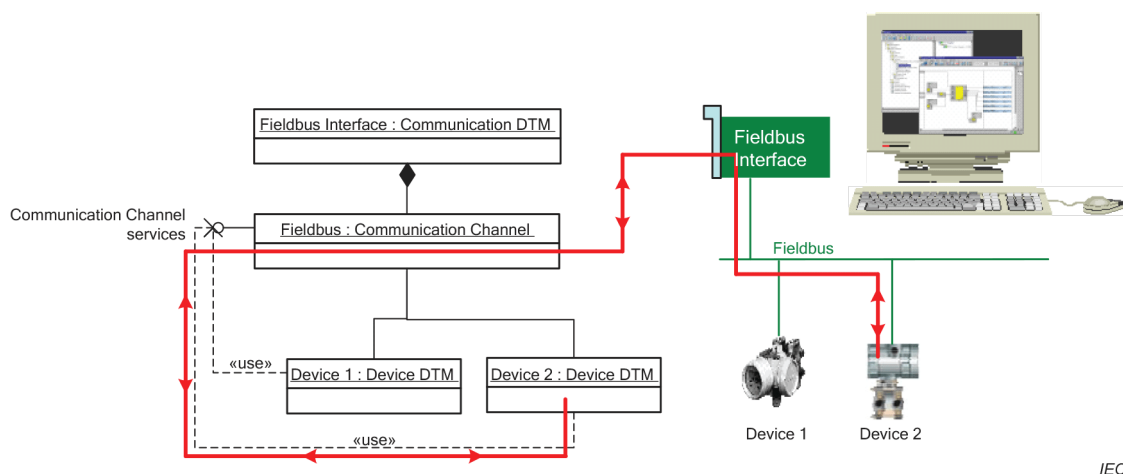


Figure 18 – Point-to-point communication

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).

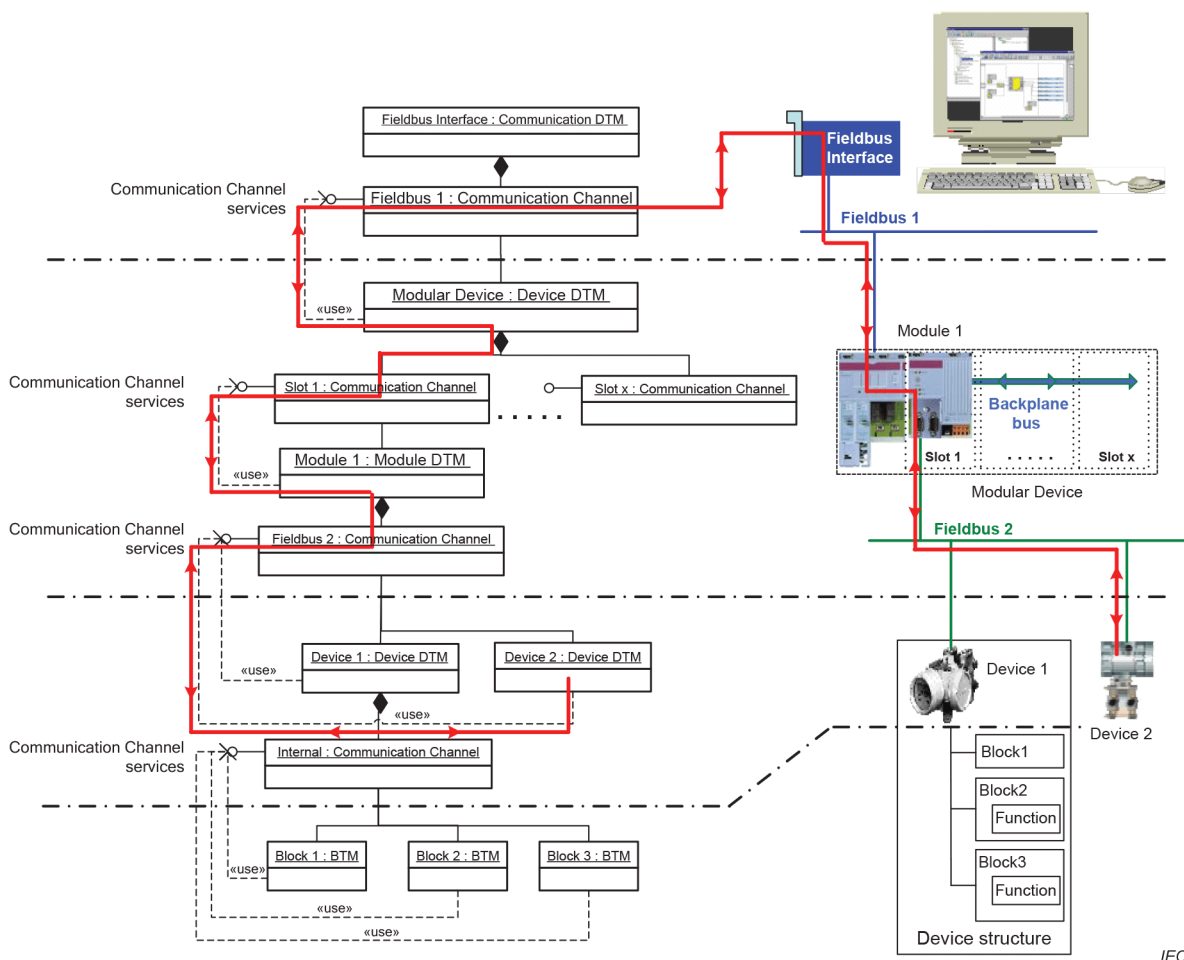


Figure 19 – Nested communication

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.

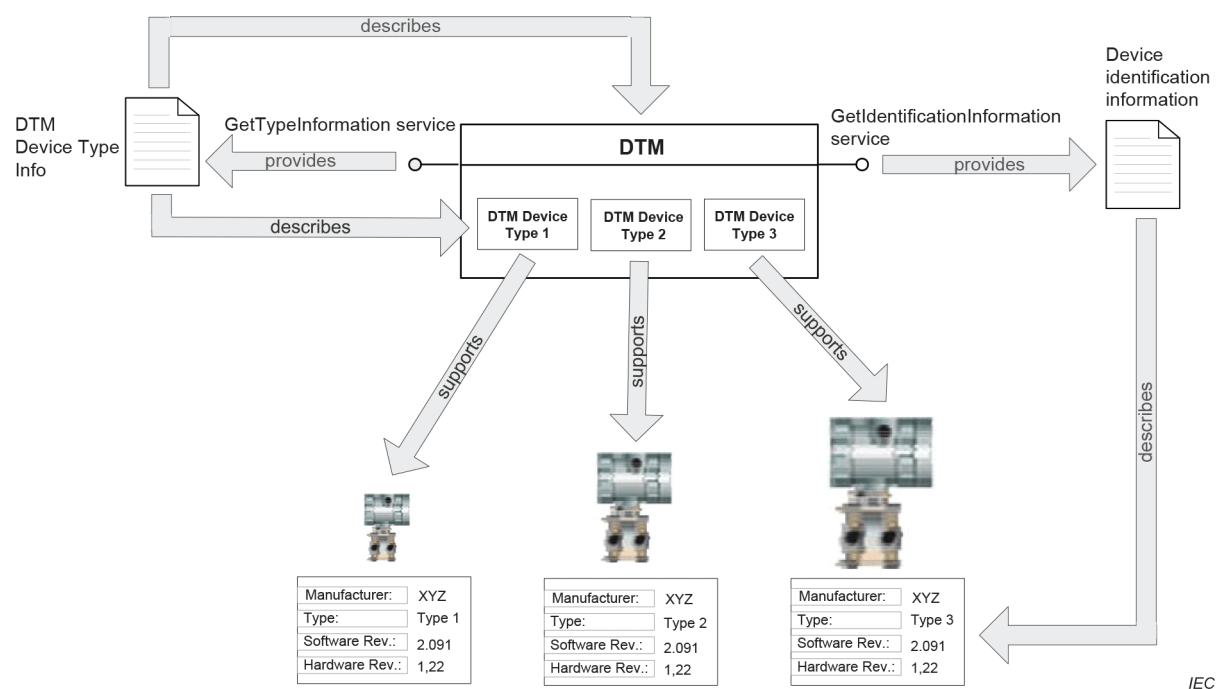


Figure 20 – DTM, DTM Device Type and device identification information

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.

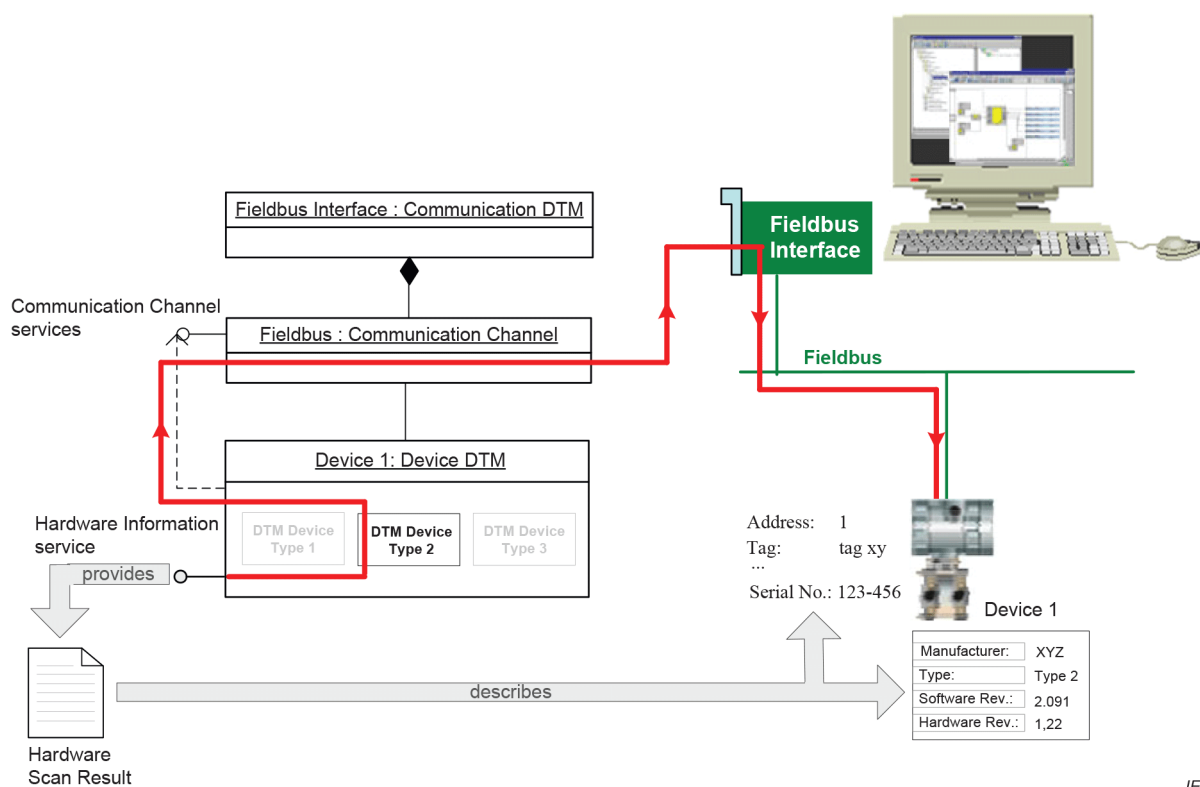
4.8.2 Supported hardware identification

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).



IEC

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).

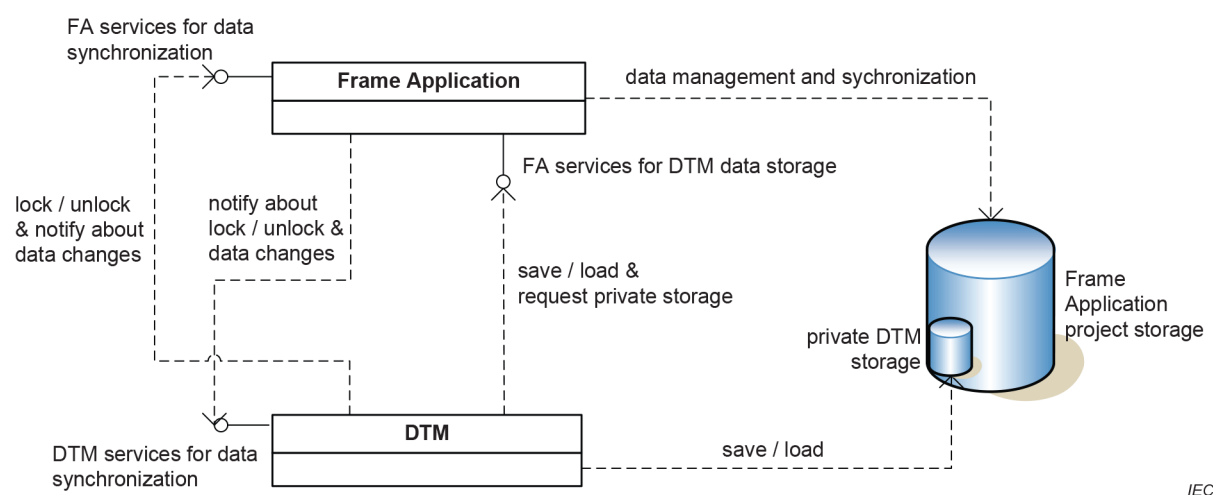


Figure 22 – FDT storage and synchronization mechanisms

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 consistency 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.

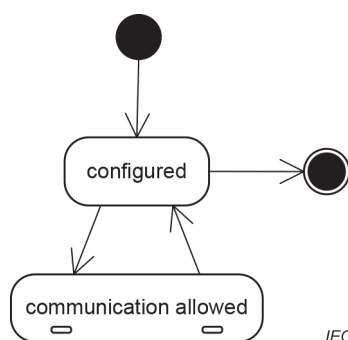


Figure 23 – DTM state machine

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.

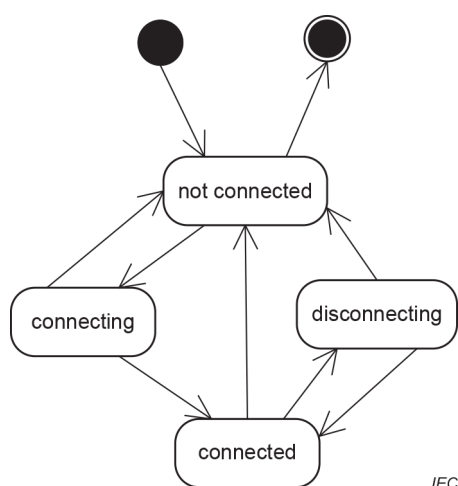
Table 3 shows an overview of these transitions.

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).



IEC

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:

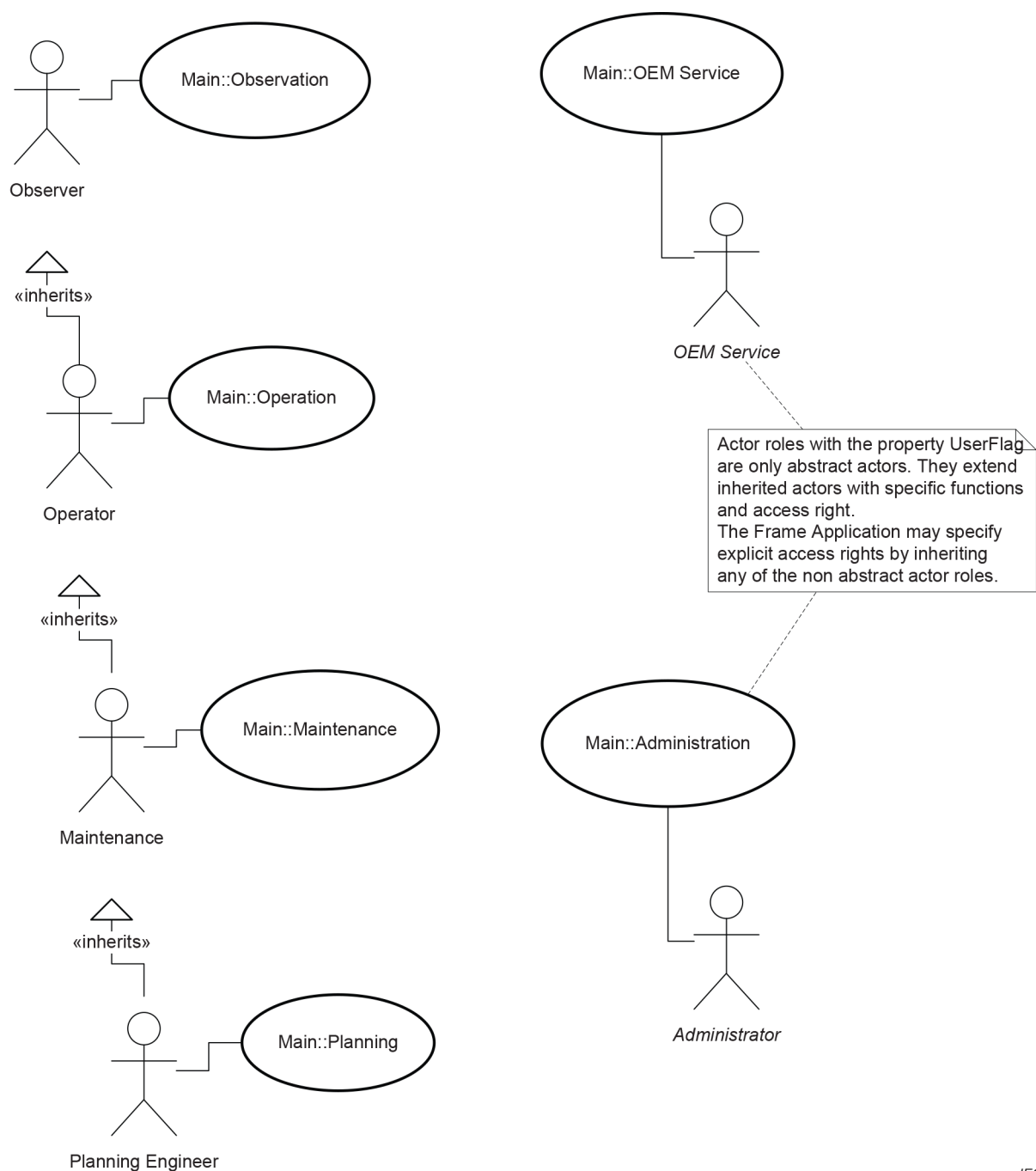
- shall return the same version as the parent and provide the functionality according to this version, or
- shall emulate the higher functionality if possible. Then the Gateway DTM shall guarantee that all required functionality is provided by its parent.

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.



IEC

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